Journal of Machine Learning Research 26 (2025) 1-46 Submitted 9/24; Revised 11/25; Published 12/25

Combining Climate Models using Bayesian Regression Trees and Random Paths

John C. Yannotty yannotty@battelle.org
Battelle Memorial Institute
Columbus, OH 43201, USA

Thomas J. Santner santner.1@osu.edu
Department of Statistics
The Ohio State University
Columbus, OH 43210, USA

Bo Li bol@wustl.edu
Department of Statistics and Data Science
Washington University in St. Louis
St. Louis, MO 63130, USA

Matthew T. Pratola mpratola@iu.edu
Department of Statistics
Indiana University
Bloomington, IN 47405, USA

Editor: Chris Oats

Abstract

General circulation models (GCMs) are essential tools for climate studies. Such climate models may have varying accuracy across the input domain, but no model is uniformly best. One can improve climate model prediction performance by integrating multiple models using input-dependent weights. Weight functions modeled using Bayesian Additive Regression Trees (BART) were recently shown to be useful in nuclear physics applications. However, a restriction of that approach was the piecewise constant weight functions. To smoothly integrate multiple climate models, we propose a new tree-based model, Random Path BART (RPBART), that incorporates random path assignments in BART to produce smooth weight functions and smooth predictions, all in a matrix-free formulation. RPBART requires a more complex prior specification, for which we introduce a semivariogram to guide hyperparameter selection. This approach is easy to interpret, computationally cheap, and avoids expensive cross-validation. Finally, we propose a posterior projection technique to enable detailed analysis of the fitted weight functions. This allows us to identify a sparse set of climate models that recovers the underlying system within a given spatial region as well as quantifying model discrepancy given the available model set. Our method is demonstrated on an ensemble of 8 GCMs modeling average monthly surface temperature.

Keywords: Bayesian Model Mixing; Climate Model Integration; Model Stacking; Soft Regression Trees

1. Introduction

Complex natural phenomena are often modeled using computer simulators – i.e. models that incorporate theoretical knowledge to approximate an underlying system. In climate applications, these simulators, or General Circulation Models (GCMs), have been widely used to understand a variety of climate features such as temperature or precipitation (Eyring et al., 2016). Many GCMs have been developed over time and each tends to have varying fidelity across different subregions of the world. Hence, no universally best model exists.

Multi-model ensembles are often used to help improve global prediction of physical systems. A common approach is to explicitly combine the outputs from K different GCMs using a linear combination or weighted average, usually in a pointwise manner for a specific latitude and longitude. For example, Knutti et al. (2017) and Tebaldi and Knutti (2007) define performance-based weights. Giorgi and Mearns (2002) define Reliability Ensemble Averaging (REA), which weight the GCMs based on model performance and model convergence. Traditional regression-based models assume the underlying process can be modeled as a linear combination of the individual GCMs (Tebaldi and Knutti, 2007). Vrac et al. (2024) propose a density-based approach that combines the individual cumulative density functions from the K models. Each of these methods estimate the weights in vastly different ways, however, they all derive location-specific weights based on the information at a given latitude and longitude location.

Alternative approaches implicitly combine GCMs to estimate the mixed-prediction. For example, Harris et al. (2023) model the underlying system as a Gaussian process with a deep neural network kernel, and then employ Gaussian process regression to combine multiple climate models. Sansom et al. (2021) define a Bayesian hierarchical model that assumes a co-exchangeable relationship between climate models and the real world process.

Global approaches such as Bayesian Model Averaging (BMA) (Raftery et al., 1997) and model stacking (Breiman, 1996; Yao et al., 2018; Clyde and Iversen, 2013), have not seen wide adoption in climate applications. These approaches use scalar weights to combine simulator outputs. The corresponding weights in such schemes are meant to reflect the overall accuracy of each model, where larger weights indicate better performance.

More recent advances consider localized weights, where the weights are explicitly modeled as functions over the input domain. The outputs from each model are then combined, or mixed, using weight functions that reflect each individual model's local predictive accuracy relative to the others in the model set. This approach is often referred to as model mixing and enables more effective learning of local information than pointwise methods without degenerating to overly simplistic global weighting. One key challenge in model mixing is specifying the relationship between the inputs and the weight values. Some specific approaches for modeling the weights use linear basis functions (Sill et al., 2009), generalized linear models (Yao et al., 2021), neural networks (Coscrato et al., 2020), calibration-based weighting (Phillips et al., 2021), Dirichlet-based weights (Kejzlar et al., 2023), precision weighting (Semposki et al., 2022), and Bayesian Additive Regression Trees (BART) (Yannotty et al., 2024). These approaches are conceptually related but differ in the assumed functional relationship for the weights and the capability to quantify uncertainties.

The BART approach for modeling the K-dimensional vector of weight functions is attractive due to its non-parametric formulation which avoids the need for user-specified basis functions. Specifically, the BART approach defines a set of prior tree bases which are adaptively learned based on the information in the model set and the observational data. However, the resulting weight functions of BART are piecewise constant, resulting in the primary drawback of this approach: the weight functions and predictions of the system are discontinuous. This is a noticeable limitation when smoothness is desirable. The univariate regression extension Soft BART (SBART) (Linero and Yang, 2018) allows smooth predictions using tree bases. However, SBART is better suited to modeling scalar responses rather than a vector-valued quantity. Additionally, Linero and Yang (2018) propose a set of default priors for the SBART model, which mitigates the need to select the values of hyperparameters and avoids a complex cross-validation study. These default settings may be insufficient in some applications and a more principled approach for prior calibration may be desired. Thus, directly applying the existing "soft" regression tree methods to the current BART-based weight functions is both not straight-forward and computationally infeasible.

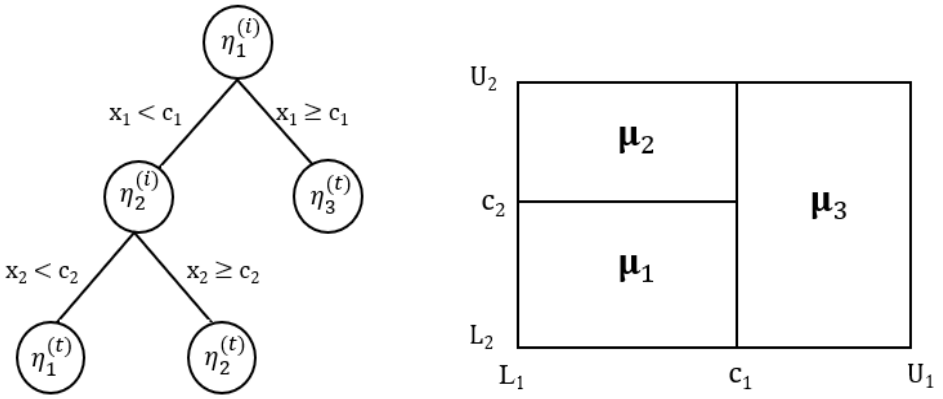
Our contributions to this research are as follows. First, we propose the Random Path BART (RPBART) model, which uses a latent variable approach to enable smooth predictions using an additive regression tree framework, all in a matrix-free formulation. Second, we introduce the random path model in the Bayesian Model Mixing framework (RPBART-BMM). The RPBART-BMM model improves on the BART-based model mixing method introduced by Yannotty et al. (2024), which was at times sensitive to overfitting and thus provided poor uncertainty quantification in areas away from the training points. The proposed construction also introduces smoothness in a holistic way such that the induced smoothing is compatible with the localization effect of the learned tree structure. Additionally, we derive the prior semivariogram of the resulting model, allowing for principled yet efficient calibration of model prior hyperparameters with similar ease as the original BART proposal. We also introduce posterior projection methods for model mixing that can be used to better interpret the mixed-prediction and resulting weight functions in cases where all, some, or none of the models are locally useful for the system of interest. Finally, we demonstrate our methodology by combining the outputs from K different GCMs to estimate the underlying mean process of the true system and gain insight as to where each GCM is locally accurate or inaccurate. We demonstrate the enhanced performance of our method relative to competing methods in mixing GCMs.

The remainder of the paper is organized as follows. Section 2 reviews the relevant background literature relating to Bayesian regression trees. Section 3 outlines our novel Random Path (RPBART) methodology for scalar responses. Section 4 extends the RPBART model to Bayesian model mixing and introduces projections of the fitted weight functions. Section 5 applies our methods to GCMs, and Section 6 summarizes our contributions.

2. Bayesian Regression Trees

Bayesian regression trees were introduced in Chipman et al. (1998, 2010); Denison et al. (1998). The most common ensemble approach is BART (Chipman et al., 2010), which models the mean function using additive tree bases.

A single tree T consists of B terminal nodes and $B-1$ internal nodes, which partitions a p-dimensional compact input space into B disjoint subsets. Each internal node encodes



(a) Example Tree

(b) Partitions of the Input Space

Figure 1: (Left) An example of a tree structure T applied to a 2-dimensional input space. The internal and terminal nodes of the tree are denoted with superscripts (i) and (t), respectively. Each internal node facilitates a binary split of the form $x_v < c_v$. (Right) The partitions of the rectangular input space $[L_1, U_1] \times [L_2, U_2]$, with associated terminal node parameters μ_b, $b = 1, 2, 3$. The function $g(\boldsymbol{x}; T, M)$ maps a given $\boldsymbol{x}$ to one of these three values.

a rule of the form $x_v < c_v$, where $v \in \{1, \ldots, p\}$, inducing a binary split along the vth dimension of the input space. Cutpoint c_v is drawn from a discretized set over the interval $[L_v, U_v]$, the lower and upper bounds of the vth dimension of the input space, respectively. The terminal nodes are found in the bottom level of the tree. Terminal node b corresponds to a unique partition of the input space and is assigned a unique terminal node parameter μ_b. Therefore, a tree implicitly defines a function $g(\boldsymbol{x}; T, M)$ such that if $\boldsymbol{x}$ lies in the bth partition, then $g(\boldsymbol{x}; T, M) = \mu_b$, where T denotes the tree topology and $M = \{\mu_1, \ldots, \mu_B\}$ denotes the set of terminal node parameters. By construction, $g(\boldsymbol{x}; T, M)$ is piecewise constant. Figure 1 displays an example with $B = 3$ terminal nodes and a 2-dimensional input space.

The tree topology prior, $\pi(T)$, defines the prior probability of node type (terminal or internal) and splitting rules for each internal node. The default prior is uninformative for the split rules and penalizes tree depth (Chipman et al., 2010) as $\pi(\eta \text{ is internal}) = \alpha(1 + d_\eta)^{-\beta}$, where α and β are tuning parameters and d_η denotes the depth of η. The output of the tree corresponds to the set of the terminal node parameters M. Typically, a conditionally independent conjugate normal prior is assigned to each $\mu_1, \ldots, \mu_B$. These assumptions simplify the Markov Chain Monte Carlo (MCMC), particularly when working with a Gaussian likelihood.

Samples from the posterior of T, M and σ^2 are generated using MCMC. The tree topology is updated at each iteration by proposing a slight change to the existing tree structure (Chipman et al., 1998; Pratola, 2016). The terminal node and variance parameters are then updated using Gibbs steps.

The methodology outlined above is easily extended to an ensemble of m trees, $T_1, \ldots, T_m$, with terminal node parameter sets $M_1, \ldots, M_m$ (Chipman et al., 2010; Hastie and Tibshirani, 2000). The mean function is then modeled as a sum-of-trees, $E[Y(\boldsymbol{x}) \mid \boldsymbol{x}] = \sum_{j=1}^{m} g(\boldsymbol{x}; T_j, M_j)$, where $g(\boldsymbol{x}; T_j, M_j) = \sum_{b=1}^{B_j} \mu_{bj}\ I(\boldsymbol{x} \in \mathcal{X}_{bj})$ and $I(\boldsymbol{x} \in \mathcal{X}_{bj})$ is an indicator function denoting the event that $\boldsymbol{x}$ maps to the bth partition of the input space by the jth tree.

Generally, tree models result in a piecewise constant mean function. When continuity is desirable, soft regression trees (Irsoy et al., 2012; Linero and Yang, 2018) are a useful alternative. A soft regression tree maps an observation with input $\boldsymbol{x}$ to a unique terminal node probabilistically given the tree topology and associated splitting rules. If T_j has B_j terminal nodes, the probability $\boldsymbol{x}$ maps to the bth terminal node is $\phi_{bj}(\boldsymbol{x}; T_j, \gamma_j)$ for $b = 1, \ldots, B_j$, for bandwidth parameter γ_j. The bandwidth γ_j controls the amount of smoothing across terminal nodes. Assuming each input dimension is standardized to $x_v \in [0, 1]$, SBART assigns an exponential prior to γ_j. Values of $\gamma_j \geq 0.1$ lead to a more global solution, while small values of γ_j generate a localized fit similar to BART. The mean function of a soft regression tree is a weighted average of the terminal node parameters

$$g(\boldsymbol{x}; T_j, M_j) = \sum_{b=1}^{B_j} \mu_{bj}\ \phi_{bj}(\boldsymbol{x}; T_j, \gamma_j). \tag{1}$$

Traditional regression trees with deterministic paths correspond to $\phi_{bj}(\boldsymbol{x}; T_j, \gamma_j) = I(\boldsymbol{x} \in \mathcal{X}_{bj})$.

The terminal node parameters in (1) need to be updated jointly during the MCMC. When these are scalars, this requires inversion of a non-sparse $B_j \times B_j$ matrix (Linero and Yang, 2018). When trees are regularized to maintain a shallow depth, this inversion is relatively inexpensive to compute. But when the parameters are K-dimensional, inverting the $KB_j \times KB_j$ matrix can become computationally expensive even with shallow trees. The proposed RPBART model alleviates these concerns.

3. The Random Path Model

We first propose our smooth, continuous, Random Path model for standard BART (RPBART), which models a univariate mean function. Let $Y(\boldsymbol{x}_i)$ be an observable quantity from some unknown process at input $\boldsymbol{x}_i$. Let $z_{bj}(\boldsymbol{x}_i)$ be the latent *random path indicator* for the event that the ith input is mapped to the bth terminal node in T_j. Given the random path assignments, we assume $Y(\boldsymbol{x}_i)$ is modeled as

$$Y(\boldsymbol{x}_i) \mid \{T_j, M_j, Z_j\}_{j=1}^{m}, \sigma^2 \ \sim\ N\Big(\sum_{j=1}^{m} g(\boldsymbol{x}_i; T_j, M_j, Z_j), \sigma^2\Big), \tag{2}$$

$$g(\boldsymbol{x}_i; T_j, M_j, Z_j) = \sum_{b=1}^{B_j} \mu_{bj}\ z_{bj}(\boldsymbol{x}_i). \tag{3}$$

Each observation is mapped to exactly one terminal node within T_j, thus $\sum_{b=1}^{B_j} z_{bj}(\boldsymbol{x}_i) = 1$ and $z_{bj}(\boldsymbol{x}_i) \in \{0,1\}$ for $b = 1,\ldots,B_j$ and $i = 1,\ldots,n$. The set Z_j is then defined as $Z_j = \{\boldsymbol{z}_j(\boldsymbol{x}_i)\}_{i=1}^n$ where $\boldsymbol{z}_j(\boldsymbol{x}_i)$ is the B_j-dimensional vector of random path assignments for the ith input. Traditional BART with deterministic paths can be viewed as a special case of RPBART where $z_{bj}(\boldsymbol{x}_i) = I(\boldsymbol{x}_i \in \mathcal{X}_{bj})$. Conditional on Z_j, the output of the random path tree remains a piecewise constant form. However, taking the expectation of the sum-of-trees with respect to Z_j results in a continuous function, similar to SBART (1).

3.1 Prior Specification

The original BART model depends on the tree structures, associated set of terminal node parameters, and error variance. We maintain the usual priors for each of these components,

$$\mu_{bj} \mid T_j \sim N(0,\tau^2), \quad T_j \sim \pi(T_j), \quad \sigma^2 \sim \nu\lambda/\chi^2_\nu,$$

where $b = 1,\ldots,B_j$ and $j = 1,\ldots,m$. The values of τ, λ, and ν can still be selected using similar methods as specified by Chipman et al. (2010).

RPBART introduces two new sets of parameters, (Z_j,γ_j) for $j = 1,\ldots,m$. For each Z_j, consider the path assignment for the ith observation, $\boldsymbol{z}_j(\boldsymbol{x}_i)$. Conditional on T_j, the B_j-dimensional random path vector for a given $\boldsymbol{x}_i$ is assigned a Multinomial prior

$$\boldsymbol{z}_j(\boldsymbol{x}_i) \mid T_j,\gamma_j \sim \text{Multinomial}\Big(1; \phi_{1j}(\boldsymbol{x}_i;T_j,\gamma_j),\ldots,\phi_{B_jj}(\boldsymbol{x}_i;T_j,\gamma_j)\Big), \tag{4}$$

where $\phi_{bj}(\boldsymbol{x}_i;T_j,\gamma_j)$ is the probability an observation with input $\boldsymbol{x}_i$ is mapped to the bth terminal node in T_j or equivalently, the conditional probability that $z_{bj}(\boldsymbol{x}_i) = 1$. The bandwidth parameter, γ_j, takes values within the interval $(0,1)$ and controls the degree of pooling across terminal nodes. As γ_j increases, more information is shared across the terminal nodes, which leads to a less localized prediction. Since we confine γ_j to the interval $(0,1)$ we assume $\gamma_j \sim \text{Beta}(\alpha_1,\alpha_2)$, for $j = 1,\ldots,m$. This prior for γ_j noticeably differs from the exponential prior in SBART (Linero and Yang, 2018). The different modeling assumptions are guided by the design of the path probabilities, $\phi_{bj}(\boldsymbol{x}_i;T_j,\gamma_j)$, which are discussed in Section 3.2. Finally, we assume conditional independence given the set of m trees which implies

$$\begin{aligned}\pi\big(\sigma^2,\{T_j,M_j,Z_j,\gamma_j\}_{j=1}^m\big) &= \pi(\sigma^2)\prod_{j=1}^m \pi(M_j \mid T_j,Z_j)\,\pi(Z_j \mid T_j,\gamma_j)\,\pi(T_j)\pi(\gamma_j) \\ &= \pi(\sigma^2)\prod_{j=1}^m \pi(T_j)\,\pi(\gamma_j)\prod_{b=1}^{B_j}\pi(\boldsymbol{\mu}_{bj} \mid T_j)\,\pi(Z_j \mid T_j,\gamma_j). \end{aligned} \tag{5}$$

Furthermore, we assume mutual independence across the random path assignment vectors $\boldsymbol{z}_j(\boldsymbol{x}_i)$ apriori. Thus the set of vectors over the n training points can be rewritten as

$$\pi(Z_j \mid T_j,\gamma_j) = \prod_{i=1}^n\prod_{b=1}^{B_j}\Big(\phi_{bj}(\boldsymbol{x}_i;T_j,\gamma_j)\Big)^{z_{bj}(\boldsymbol{x}_i)}. \tag{6}$$

3.2 The Path Probabilities

The continuity in the mean prediction from each tree is driven by the B_j path probabilities, $\phi_{bj}(\boldsymbol{x}_i; T_j, \gamma_j)$. Recall, a tree model recursively partitions the input space into B_j disjoint subregions using a sequence of splitting rules. We define the path from the root node, $\eta_{1j}^{(i)}$, to the bth terminal node, $\eta_{bj}^{(t)}$, in terms of the sequence of internal nodes that connect $\eta_{1j}^{(i)}$ and $\eta_{bj}^{(t)}$. To define $\phi_{bj}(\boldsymbol{x}_i; T_j, \gamma_j)$, we must consider the probability of visiting the internal nodes that form the path that connects $\eta_{1j}^{(i)}$ and $\eta_{bj}^{(t)}$.

Consider the tree with $B_j = 3$ terminal nodes (red, blue, green) and induced partition of the 1-dimensional input space, $[-1, 1]$, in Figure 3. The path to $\eta_{1j}^{(t)}$ (red) is defined by splitting left at the root node. Similarly, the path $\eta_{2j}^{(t)}$ (blue) is defined by splitting right at $\eta_{1j}^{(i)}$ and then left at $\eta_{2j}^{(i)}$. In the usual tree model, these splits happen deterministically. This means an observation with $x_1 < 0$ splits left and is thus mapped to $\eta_{1j}^{(t)}$ with probability 1. With random paths, an observation splits right with probability $\psi(\boldsymbol{x}; \cdot)$ and left with probability $1 - \psi(\boldsymbol{x}; \cdot)$. This adds another layer of stochasticity into the model.

3.2.1 Defining the Splitting Probabilities

Consider the dth internal node, $\eta_{dj}^{(i)}$, of T_j. Assume $\eta_{dj}^{(i)}$ splits on the rule $x_{v_{(dj)}} < c_{(dj)}$. Let $\psi(\boldsymbol{x}; v_{(dj)}, c_{(dj)}, \gamma_j)$ define the probability an observation with input $\boldsymbol{x}$ moves to the right child of $\eta_{dj}^{(i)}$. Further assume the cutpoint $c_{(dj)}$ is selected from the discretized subset $(L_v^d, U_v^d) \cap \mathcal{C}_v$, where $\mathcal{C}_v$ is the finite set of possible cutpoints for variable v, and L_v^d and U_v^d are the upper and lower bounds defined based on the previous splitting rules in the tree. The bounds, which are computed based on the information in T_j, are used to define a threshold which establishes a notion of "closeness" between points. We incorporate this information into the definition of $\psi(\boldsymbol{x}; v_{(dj)}, c_{(dj)}, \gamma_j)$ by

$$\psi(\boldsymbol{x}; v_{(dj)}, c_{(dj)}, \gamma_j) = \begin{cases} 1 - \frac{1}{2}\left(1 - \frac{x_{v_{(dj)}} - c_{(dj)}}{\gamma_j (U_v^d - c_{(dj)})}\right)_+^q & x_{v_{(dj)}} \geq c_{(dj)}, \\ \frac{1}{2}\left(1 - \frac{c_{(dj)} - x_{v_{(dj)}}}{\gamma_j (c_{(dj)} - L_v^d)}\right)_+^q & x_{v_{(dj)}} < c_{(dj)}, \end{cases} \tag{7}$$

where the expression $a_+ = \max\{a, 0\}$ for any $a \in \mathbb{R}$ and q is a shape parameter. This definition of $\psi(\boldsymbol{x}; v_{(dj)}, c_{(dj)}, \gamma_j)$ restricts the probabilistic assignment to the left or right child of $\eta_{dj}^{(i)}$ to the interval $\mathcal{I}_{dj}(\gamma_j) := \big(c_{(dj)} - \gamma_j(c_{(dj)} - L_v^d),\ c_{(dj)} + \gamma_j(U_v^d - c_{(dj)})\big)$. Any observation with input $x_{v_{(dj)}}$ such that $x_{v_{(dj)}} \in \mathcal{I}_{dj}(\gamma_j)$ has a non-zero chance of being assigned to either of the child nodes in the binary split. Meanwhile, $\psi(\boldsymbol{x}; v_{(dj)}, c_{(dj)}, \gamma_j) = 0$ if $x_{v_{(dj)}}$ is less than the lower bound in $\mathcal{I}_{dj}(\gamma_j)$ or $\psi(\boldsymbol{x}; v_{(dj)}, c_{(dj)}, \gamma_j) = 1$ if $x_{v_{(dj)}}$ is greater than the upper bound in $\mathcal{I}_{dj}(\gamma_j)$. This means the probabilistic assignment agrees with the deterministic split when $x_{v_{(dj)}} \notin \mathcal{I}_{dj}(\gamma_j)$. In other words, points farther away from the cutpoint deterministically move to the left or right, while points close to the cutpoint take a random move left or right.

To understand γ_j, consider the first split (at the root node) within a given a tree. Assume the root node splits the interval $[L_1^{(1)}, U_1^{(1)}] = [-1, 1]$ using the rule $x_1 < 0$ (i.e. $v_{1j} = 1$ and

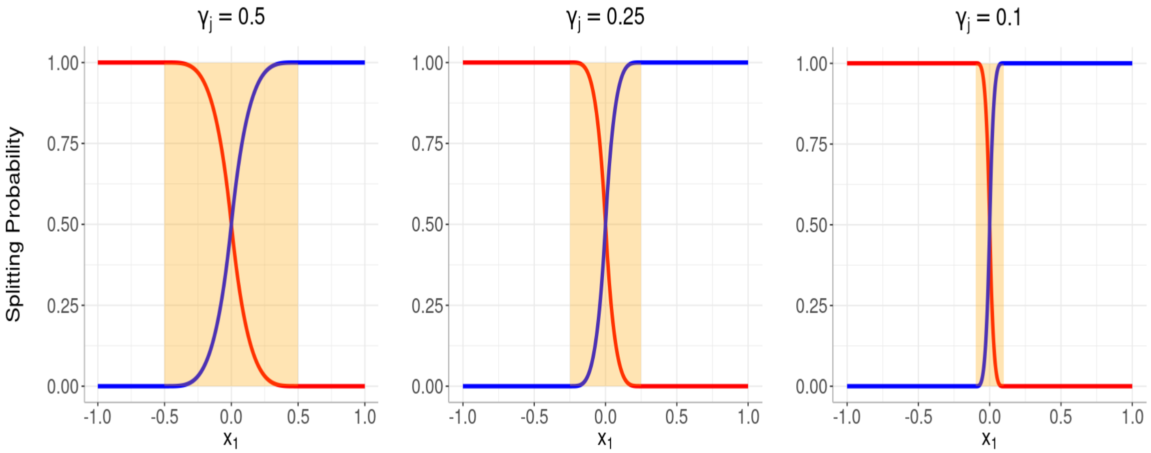


Figure 2: Three examples of $1-\psi(\boldsymbol{x};1,0,\gamma_j)$ (red) and $\psi(\boldsymbol{x};1,0,\gamma_j)$ (blue) with bandwidth parameters of 0.5, 0.25, and 0.1, corresponding to the move that splits the interval $[L_1^{(1)}, U_1^{(1)}] = [-1,1]$ on the cutpoint $c_{1j} = 0$. The interval $\mathcal{I}_{1j}(\gamma_j)$ (orange) defines the interval where the random paths can disagree with the traditional deterministic paths.

$c_{1j} = 0$). Figure 2 displays the probability of splitting left (red), $1-\psi(\boldsymbol{x};1,0,\gamma_j)$, and the probability of splitting right (blue), $\psi(\boldsymbol{x};1,0,\gamma_j)$, for different values of γ_j as a function of x_1. The orange region in each panel highlights the set $\mathcal{I}_{1j}(\gamma_j)$. The interval is wider for larger γ_j, which means a larger proportion of points can split left or right. Lower γ_j results in steeper curves and $\psi(\boldsymbol{x};1,0,\gamma_j)$ starts to resemble the deterministic rule $I(x_{v_{(dj)}} \geq c_{(dj)})$.

The splitting probabilities determine how to traverse the tree along the various paths created by the internal nodes. These probabilities set the foundation for computing the probability of reaching any of the B_j terminal nodes within T_j.

3.2.2 Defining the Path Probabilities

The path probabilities, $\phi_{bj}(\boldsymbol{x};T_j,\gamma_j)$, can be defined in terms of the individual splits at each internal node. Consider the tree with $B_j = 3$ terminal nodes in Figure 3. The first internal node, $\eta_{1j}^{(i)}$, splits using $v_{1j} = 1$ and $c_{1j} = 0$, while the second internal node, $\eta_{2j}^{(i)}$, splits using $v_{2j} = 1$ and $c_{2j} = 0.4$. The probability of reaching $\eta_{1j}^{(t)}$ is simply the probability of splitting left at $\eta_{1j}^{(i)}$, which is given by $\phi_{1j}(\boldsymbol{x};T_j,\gamma_j) = 1-\psi(\boldsymbol{x};1,0,\gamma_j)$. Meanwhile, an observation is mapped to the blue terminal node by first splitting right at $\eta_{1j}^{(i)}$ and then left at $\eta_{2j}^{(i)}$. This occurs with probability $\phi_{2j}(\boldsymbol{x};T_j,\gamma_j) = \psi(\boldsymbol{x};1,0,\gamma_j) \times \big(1-\psi(\boldsymbol{x};1,0.4,\gamma_j)\big)$. Figure 3 displays the resulting path probabilities for each terminal node with $\gamma_j = 0.5$.

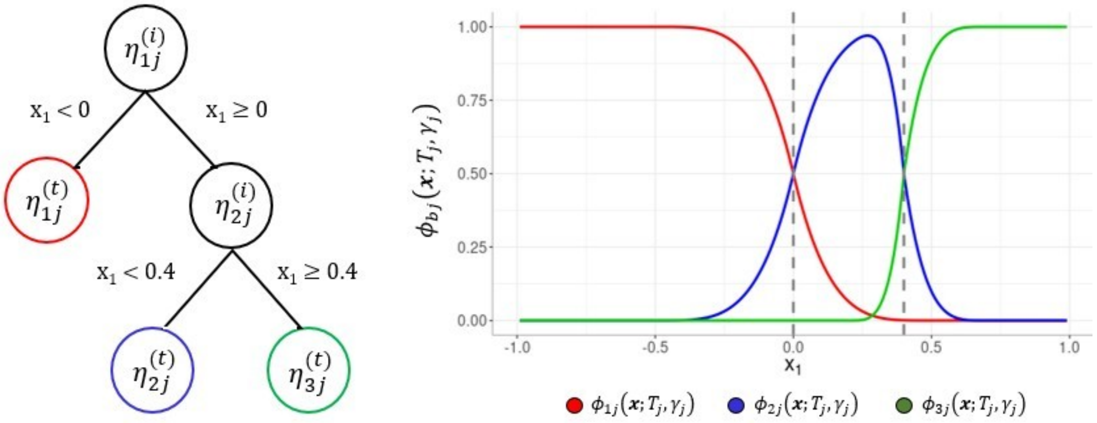

$\eta_{1j}^{(i)}$
$x_1 < 0$
$x_1 \geq 0$
$\eta_{1j}^{(t)}$
$\eta_{2j}^{(i)}$
$x_1 < 0.4$
$x_1 \geq 0.4$
$\eta_{2j}^{(t)}$
$\eta_{3j}^{(t)}$
$\phi_{bj}(\boldsymbol{x}; T_j, \gamma_j)$
1.00
0.75
0.50
0.25
0.00
-1.0
-0.5
0.0
0.5
1.0
x_1
● $\phi_{1j}(\boldsymbol{x}; T_j, \gamma_j)$
● $\phi_{2j}(\boldsymbol{x}; T_j, \gamma_j)$
● $\phi_{3j}(\boldsymbol{x}; T_j, \gamma_j)$

Figure 3: (Left) An example tree with $B_j = 3$ terminal nodes (red, blue, green). The internal nodes (black) define a set of splitting rules that recursively partition the input space. (Right) Example path probabilities, $\phi_{bj}(\boldsymbol{x}; T_j, \gamma_j)$, using $\gamma_j = 0.5$. Each curve displays the probability of being mapped to the corresponding terminal node as a function of x_1.

In general, if the path from the root node to the bth terminal node depends on D internal nodes, the path probability is defined by

$$\phi_{bj}(\boldsymbol{x}; T_j, \gamma_j) = \prod_{d=1}^{D} \psi(\boldsymbol{x}; v_{(dj)}, c_{(dj)}, \gamma_j)^{R_{(dj)}} \times \big(1 - \psi(\boldsymbol{x}; v_{(dj)}, c_{(dj)}, \gamma_j)\big)^{1-R_{(dj)}}, \tag{8}$$

where $v_{(dj)}$ and $c_{(dj)}$ are the variable and cutpoint selected at the dth internal node along the specified path in T_j, and $R_{(dj)} = 1$ ($R_{(dj)} = 0$) if a right (left) move is required at the dth internal node to continue along the path towards $\eta_{bj}^{(t)}$.

3.3 Smooth Mean Predictions

Let $Y(\tilde{\boldsymbol{x}})$ be a new observable quantity at input $\tilde{\boldsymbol{x}}$. Conditional on the m random path vectors at $\tilde{x}$, the mean function is given by

$$E[Y(\tilde{\boldsymbol{x}}) \mid \{T_j, M_j, \tilde{Z}_j, \gamma_j\}_{j=1}^{m}] = \sum_{j=1}^{m} \sum_{b=1}^{B_j} \boldsymbol{\mu}_{bj} \, z_{bj}(\tilde{\boldsymbol{x}}),$$

where $\tilde{Z}_j = Z_j \cup \{\boldsymbol{z}_j(\tilde{\boldsymbol{x}})\}$ is the set of random path assignments from the training data and the future observation. Marginalizing over $\tilde{Z}_j$ results in a smooth mean prediction,

$$E[Y(\tilde{\boldsymbol{x}}) \mid \{T_j, M_j, \gamma_j\}_{j=1}^{m}] = \sum_{j=1}^{m} \sum_{b=1}^{B_j} \boldsymbol{\mu}_{bj} \, \phi_{bj}(\tilde{\boldsymbol{x}}; T_j, \gamma_j). \tag{9}$$

Posterior samples of this expectation can be obtained by evaluating the functional form in (9) given posterior draws of T_j, M_j, and γ_j.

3.4 The Semivariogram

The RPBART model introduces an additional pair of hyperparameters, α_1 and α_2, which control the m bandwidth parameters γ_j, and therefore the level of smoothing in the model. The common approach in BART is to calibrate the prior hyperparameters using a lightly data-informed approach or cross validation. However, neither the data-informed approach proposed for BART (Chipman et al., 2010) nor the default prior settings in SBART (Linero and Yang, 2018) can be used for calibrating the new smoothness parameters α_1 and α_2. Furthermore, the added complexity of RPBART would render the traditional cross-validation study too complex and computationally expensive.

We propose to calibrate the RPBART model using the semivariogram (Cressie, 2015). The semivariogram, $\overline{\nu}(\|\boldsymbol{h}\|)$, is defined by

$$\overline{\nu}(\|\boldsymbol{h}\|) = \frac{1}{|\mathcal{X}|}\int_{\mathcal{X}} \nu(\boldsymbol{x},\boldsymbol{h})\, d\boldsymbol{x}, \tag{10}$$

$$\nu(\boldsymbol{x},\boldsymbol{h}) = \frac{1}{2}\mathrm{Var}\big(Y(\boldsymbol{x}+\boldsymbol{h}) - Y(\boldsymbol{x})\big), \tag{11}$$

where $|\mathcal{X}|$ denotes the volume of the input domain $\mathcal{X}$ and $\boldsymbol{x}+\boldsymbol{h}$ denotes an input which is a distance $\|\boldsymbol{h}\| > 0$ away from $\boldsymbol{x}$ (Matheron, 1963). Assuming a constant mean for $Y(\boldsymbol{x})$, the function $\nu(\boldsymbol{x},\boldsymbol{h})$ simplifies as $\nu(\boldsymbol{x},\boldsymbol{h}) = \frac{1}{2}E\Big[\big(Y(\boldsymbol{x}+\boldsymbol{h}) - Y(\boldsymbol{x})\big)^2\Big]$. The function $\nu(\boldsymbol{x},\boldsymbol{h})$ describes the spatial correlation between two points that are separated by a distance of $\|\boldsymbol{h}\|$ and may depend on parameters that determine the shape of the semivariogram $\overline{\nu}(\|\boldsymbol{h}\|)$.

We propose calibrating the hyperparameters of the RPBART model by using an estimator of the prior semivariogram in (10) and the function $\nu(\boldsymbol{x},\boldsymbol{h})$ in (11). The estimator of (11) can be calculated by marginalizing over the set of parameters $\{T_j, M_j, Z_j, \gamma_j\}_{j=1}^m$ and conditioning on σ^2. In order to compute $\nu(\boldsymbol{x},\boldsymbol{h})$, we first analytically marginalize over $\{M_j, Z_j\}_{j=1}^m$ conditional on the remaining parameters. An expression for $\nu(\boldsymbol{x},\boldsymbol{h})$ is then computed by numerically integrating over $\{T_j, \gamma_j\}_{j=1}^m$ (where σ^2 is held fixed). The function $\nu(\boldsymbol{x},\boldsymbol{h})$ for the RPBART model is given by Theorem 3.1.

Theorem 3.1 *Assume the random quantities $\{T_j, M_j, Z_j, \gamma_j\}_{j=1}^m$ are distributed as specified in Section 3.1. Conditional on σ^2, the function $\nu(\boldsymbol{x},\boldsymbol{h})$ for the RPBART model is*

$$\nu(\boldsymbol{x},\boldsymbol{h}) = \sigma^2 + m\tau^2\Big(1 - \bar{\Phi}(\boldsymbol{x},\boldsymbol{h})\Big), \tag{12}$$

where $m\tau^2 = \Big(\frac{y_{max}-y_{min}}{2k}\Big)^2$ defines the variance of the sum-of-trees, k is a tuning parameter, $y_{max} - y_{min}$ is the range of the observed data, and

$$\bar{\Phi}(\boldsymbol{x},\boldsymbol{h}) = E\Big[\sum_{b=1}^{B_1} \phi_{b1}(\boldsymbol{x}+\boldsymbol{h};\gamma_1,T_1)\phi_{b1}(\boldsymbol{x};\gamma_1,T_1)\Big]$$

is the probability that two observations with inputs $\boldsymbol{x}$ and $\boldsymbol{x}+\boldsymbol{h}$ are assigned to the same partition. Without loss of generality, the expectation is with respect to T_1 and γ_1 (since the m trees are a priori i.i.d.).

The proof of Theorem 3.1 is in Section 7.2 of the Supplement. The expectation in Theorem 3.1, $\bar{\Phi}(\boldsymbol{x},\boldsymbol{h})$, can be approximated using draws from the prior. Thus, $\nu(\boldsymbol{x},\boldsymbol{h})$ depends on the probability that two points are assigned to different partitions, averaged over the set of trees and bandwidths, as denoted by $1-\bar{\Phi}(\boldsymbol{x},\boldsymbol{h})$. This probability is then scaled and shifted by the variance of the sum-of-trees, $m\tau^2$, and error variance, σ^2.

Generally, we are more interested in $\overline{\nu}(\|\boldsymbol{h}\|)$, which describes the variability across the entire input space rather than at a specific $\boldsymbol{x}$. For RPBART, we can numerically compute $\overline{\nu}(\|\boldsymbol{h}\|)$ by integrating (12) over $\mathcal{X}$ as shown in (10). We can use the resulting a priori semivariogram estimator to guide the selection of the hyperparameters, α, β, α_1, α_2, and k, across the various priors in the model. Note the number of trees m must still be selected through other means, such as cross-validation.

We note this procedure for computing the semivariogram is more complex than what is observed with Gaussian Processes (Cressie, 2015). In such cases, shift-invariant kernels are typically selected for the covariance model in a stationary Gaussian Process. In other words, the covariance model is simply a function of $\boldsymbol{h}$ under a shift-invariant kernel. However, the RPBART covariance function, $\bar{\Phi}(\boldsymbol{x},\boldsymbol{h})$, is a non-shift-invariant product kernel of the path probabilities that is averaged over all possible trees and bandwidth parameters. Because of this, we must consider the computation of $\nu(\boldsymbol{x},\boldsymbol{h})$ in order to numerically compute $\overline{\nu}(\|\boldsymbol{h}\|)$.

For example, consider the semivariogram with respect to the 2-dimensional input space $[-1,1]\times[-1,1]$. Figure 4 displays possible semivariograms for different values of k, α_1, and α_2 with $y_{min}=-1$ and $y_{max}=1$. Note, we set the nugget $\sigma=0$ in this example to clearly isolate the effect of the smoothness parameters α_1 and α_2, i.e. ignoring the nugget. Each panel corresponds to different settings of the bandwidth prior, where $\alpha_1=2$ and $\alpha_2=25$ indicate low levels of smoothing and $\alpha_1=10$ and $\alpha_2=5$ indicate high levels of smoothing. Based on each curve, we see the smoothing parameters primarily affect the behavior of the semivariogram for small values of $\|\boldsymbol{h}\|$. With less smoothing (left), each of the semivariograms take values closer to 0 when $\|\boldsymbol{h}\|$ is small, while a noticeable shift upwards is observed with more smoothing (center, right). This offset appears in the RPBART semivariogram regardless of the γ_j prior because the covariance function is discontinuous.

As $\|\boldsymbol{h}\|$ increases, each semivariogram reaches a maximum value, known as the sill. For each k, we observe the sill is similar regardless of the values of α_1 and α_2. Thus, the value of k controls the height of the sill and in turn the amount of variability attributed to the sum-of-trees model. As a result, the interpretation of k under the RPBART model is the same as in the original BART model. Finally, we should note that the hyperparameters in the tree prior, α and β, can also be determined based on the semivariogram, as deeper trees result in a larger number of partitions and thus less correlation across the input space. Hence, deeper trees shift the semivariograms upwards and further contribute to the offset.

In practice, we can compare the theoretical semivariogram to the empirical semivariogram to select the hyperparameters for the model (Cressie, 2015). In summary, we see α_1 and α_2 primarily affect the semivariogram when $\|\boldsymbol{h}\|$ is near 0, as more smoothing generally leads to an upward shift and increase in curvature near the origin. The tree prior hyper-

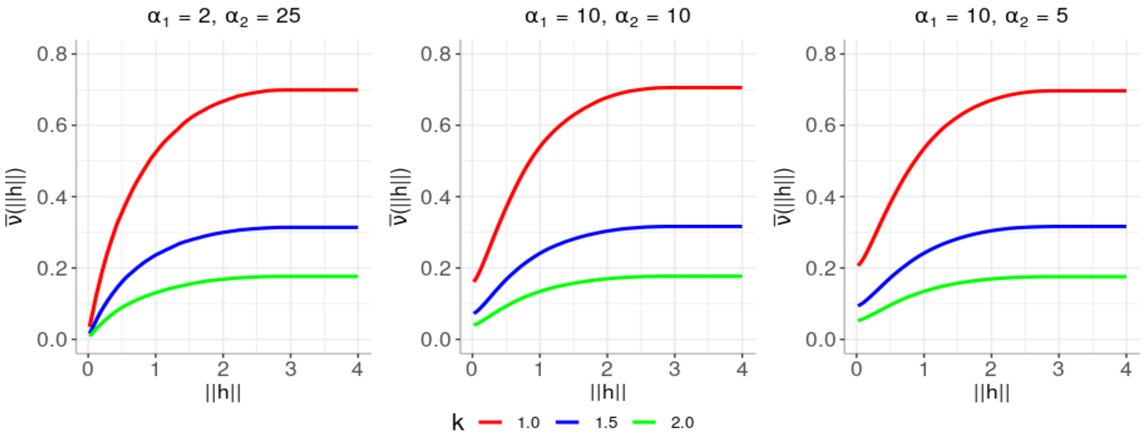


Figure 4: The semivariogram for different values of $k = 1$ (red), $k = 1.5$ (blue), and $k = 2$ (green) and bandwidth hyperparameters α_1 and α_2. Each semivariogram is generated using $y_{min} = -1$, $y_{max} = 1$, $\sigma = 0$, and $\alpha = 0.95$, and $\beta = 0.5$.

parameters, α and β, affect the size of the offset and the height of the sill. Meanwhile, the value of k simply scales the function and has minimal impact on the overall shape of the curve.

The last component to consider is the value of σ^2, which is set to 0 in Figure 4. The typical way to calibrate the $\nu\lambda/\chi^2_\nu$ prior on σ^2 is to fix ν at a desired value and select λ based on an estimate of σ^2, denoted by $\hat{\sigma}^2$ (Chipman et al., 2010; Linero and Yang, 2018). Traditional approaches typically compute $\hat{\sigma}^2$ using a linear model. From Theorem 3.1, we see σ^2 simply adds to the offset by vertically shifting the semivariogram. Thus, we can select $\hat{\sigma}^2$ based on theoretical and empirical semivariogram. Given a value of ν, one can set $\hat{\sigma}^2$ as the mode of the prior distribution and algebraically solve for λ. This strategy selects λ along with the other hyperparameters using the information encoded in the semivariogram.

Additional examples of the RPBART semivariogram, including a real world application to climate models, can be found in the supplementary material.

4. Smooth Model Mixing

4.1 A Mean-Mixing Approach

The RPBART model easily extends to the model mixing framework presented by Yannotty et al. (2024). Let $f_1(\boldsymbol{x}_i), \ldots, f_K(\boldsymbol{x}_i)$ denote the output from K simulators at $\boldsymbol{x}_i$. When the simulators are computationally expensive, the output $f_l(\boldsymbol{x}_i)$ is replaced with the prediction from an inexpensive emulator, $\hat{f}_l(\boldsymbol{x}_i)$, for $l = 1, \ldots, K$. For example, $\hat{f}_l(\boldsymbol{x}_i)$ could be the mean prediction from a Gaussian process (Santner et al., 2018) or an RPBART emulator. In climate applications, each GCM may be evaluated on different latitude and longitude grids, thus we use regridding techniques such as bilinear interpolation to compute $\hat{f}_l(\boldsymbol{x}_i)$.

Given the mean predictions from K emulators, we assume $Y(\boldsymbol{x}_i)$ is modeled by

$$Y(\boldsymbol{x}_i) \mid \hat{\boldsymbol{f}}(\boldsymbol{x}_i), \{T_j, M_j, Z_j\}_{j=1}^m, \sigma^2 \ \sim\ N\big(\hat{\boldsymbol{f}}^\top(\boldsymbol{x}_i)\boldsymbol{w}(\boldsymbol{x}_i), \sigma^2\big),$$

$$\boldsymbol{w}(\boldsymbol{x}_i) = \sum_{j=1}^{m}\sum_{b=1}^{B_j} \boldsymbol{\mu}_{bj} z_{bj}(\boldsymbol{x}_i),$$

where $\hat{\boldsymbol{f}}(\boldsymbol{x}_i)$ is the K-dimensional vector of mean predictions at input $\boldsymbol{x}_i$, $\boldsymbol{w}(\boldsymbol{x}_i)$ is the corresponding K-dimensional weight vector, and $i = 1, \ldots, n$. This extension allows the K weights to be modeled as continuous functions using similar arguments as outlined for the 1-dimensional tree output case in Section 3.3.

Similar to Yannotty et al. (2024), we regularize the weight functions via a prior on the terminal node parameters. The primary goal is to ensure each weight function, $w_l(\boldsymbol{x})$, prefers the interval $[0, 1]$ without directly imposing a strict non-negativity or sum-to-one constraint. Thus, we assume $\boldsymbol{\mu}_{bj} \mid T_j \sim N\Big(\frac{1}{mK}\mathbf{1}_K\ , \tau^2 I_K\Big)$, where $\tau = 1/(2k\sqrt{m})$ and k is a tuning parameter. This choice of τ follows Chipman et al. (2010) in that the confidence interval for the sum-of-trees has a length of $w_{max} - w_{min}$, where 0 and 1 are the target minimum and maximum values of the weights. This prior calibration ensures each $w_l(\boldsymbol{x})$ is centered about $1/K$, implying each model is equally weighted at each $\boldsymbol{x}$ apriori. The value of k controls the flexibility of the weight functions. Small k allows for flexible weights that can easily vary beyond the target bounds of 0 and 1 and thus identify granular patterns in the data. Larger k will keep the weights near the simple average of $1/K$ and could be limited in identifying regional patterns in the data. Typically, more flexible weights are needed when mixing lower fidelity models, as the weights will account for any discrepancy between the model set and the observed data. By default, we set $k = 1$.

RPBART allows for independent sampling of each of the B_j terminal node parameters, $\boldsymbol{\mu}_{bj}$, within T_j for $j = 1, \ldots, m$. Thus, RPBART avoids the joint update required in SBART. This is rather significant, as the SBART update for the B_j terminal node parameters vectors would require an inversion of a $KB_j \times KB_j$ matrix. In larger scale problems with larger K or deeper trees, the repeated inversion cost of this matrix would be burdensome.

Prior calibration of the weight functions can still be informed using the semivariogram approach. Section 7.4 provides the formula for the RPBART-BMM semivariogram along with the derivation of the formula.

In general, RPBART-BMM model introduces an additional layer of randomness into the BART-BMM framework. In BART-BMM, final mixed solution tended to overfit the data, particularly in regions where no model under consideration provided a high-fidelity approximation of the underlying system. In such regions of the domain, BART-BMM would simply find a useful combination of the models to best fit the data. The corresponding tree models would then exhibit poor mixing within this partition of the domain. The random path concept helps alleviate some of the overfitting, as training points can be mapped to adjacent terminal nodes at each iteration of the MCMC, even if the sampled tree structure remains unchanged.

4.2 Posterior Weight Projections

The RPBART weights are modeled as unconstrained functions of the inputs. Alternative approaches impose constraints on the weight functions, such as a non-negativity or sum-to-one constraint (Yao et al., 2021; Coscrato et al., 2020; Breiman, 1996). Although constrained approaches may introduce bias into the mean of the mixed prediction, they can improve the interpretability of the weight functions. However, such constraints could increase the computational complexity of the model. For example, a simplex constraint on the RPBART-based weights would drastically change the model fitting procedure, as the conditional conjugacy through the multivariate normal prior would likely be lost.

Rather than changing the model and estimation procedure, one alternative is to explore the desired constraints through post-processing methods which impose constraints a posteriori rather than a priori (Lin and Dunson, 2014; Sen et al., 2018). In this regard, we define a constrained model in terms of the original unconstrained model using Definition 1.

Definition 1 *Let $\boldsymbol{w}(\boldsymbol{x}) = \big(w_1(\boldsymbol{x}), \ldots, w_K(\boldsymbol{x})\big)^\top$ be a vector of unconstrained weights and $Y(\boldsymbol{x})$ be an observable quantity from the underlying physical process. Define the unconstrained and constrained models for $Y(\boldsymbol{x})$ by*

$$\textit{(Unconstrained)} \quad Y(\boldsymbol{x}) = \sum_{l=1}^{K} w_l(\boldsymbol{x})\hat{f}_l(\boldsymbol{x}) + \epsilon(\boldsymbol{x}),$$
$$\textit{(Constrained)} \quad Y(\boldsymbol{x}) = \sum_{l=1}^{K} u_l(\boldsymbol{x})\hat{f}_l(\boldsymbol{x}) + \delta(\boldsymbol{x}) + \epsilon(\boldsymbol{x}),$$

where $\boldsymbol{u}(\boldsymbol{x}) = \big(u_1(\boldsymbol{x}), \ldots, u_K(\boldsymbol{x})\big)^\top$ is the projection of $\boldsymbol{w}(\boldsymbol{x})$ onto the constrained space, and

$$\delta(\boldsymbol{x}) = \sum_{l=1}^{K} \big(w_l(\boldsymbol{x}) - u_l(\boldsymbol{x})\big)\hat{f}_l(\boldsymbol{x})$$

denotes the estimated discrepancy between the constrained mixture of simulators and the underlying process, and $\epsilon(\boldsymbol{x})$ is a random error.

This framework allows for the interpretation of the weight functions on the desired constrained space without introducing significant computational burdens. The unconstrained and constrained models are connected through an additive discrepancy, $\delta(\boldsymbol{x})$, which accounts for any potential bias introduced by the set of constraints.

Posterior samples of $\boldsymbol{u}(\boldsymbol{x})$ can easily be obtained by projecting the posterior samples of $\boldsymbol{w}(\boldsymbol{x})$ onto the constrained space. The specific form of the projection will depend on the desired constraints. In this work, we explore projections that enforce a simplex constraint that result in continuous constrained weight functions, are computationally cheap, and promote some level of sparsity. The simplex constraint ensures each $u_l(\boldsymbol{x}) \geq 0$ and $\sum_{l=1}^{K} u_l(\boldsymbol{x}) = 1$, while also enabling a more clear interpretation of the weights, as the mixed-prediction is simply an interpolation between different model predictions. The simplex constraint can be enforced by defining $\boldsymbol{u}(\boldsymbol{x})$ as a function of $\boldsymbol{w}(\boldsymbol{x})$ via a softmax or penalized L_2 projection (Laha et al., 2018; Kong et al., 2020). These projections each have closed form expres-

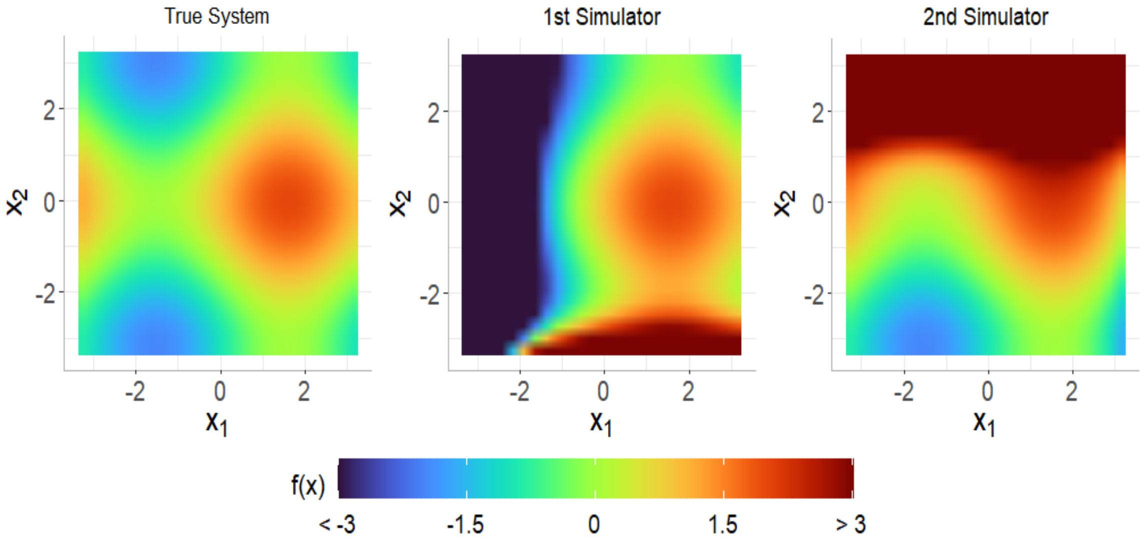


Figure 5: (Left) The underlying system $f_\dagger(\boldsymbol{x})$. (Center) The output of the first simulator. (Right) The output of the second simulator.

sions and are inexpensive to compute. One can choose between these two methods (among others) based on the desirable properties in the constrained weights and discrepancy.

The softmax is used in stacking methods such as Bayesian Hierarchical Stacking (Yao et al., 2021). Although the softmax function is widely used, a common criticism is the lack of sparsity in that the constrained weights will take values between 0 and 1 but never exactly reach either bound. Thus, the softmax can shrink the effect of a given model, but each of the K models will have at least some non-zero contribution to the final prediction. The penalized L_2 projection (i.e. the sparsegen-linear from Laha et al. (2018)) defines the weights using a thresholding function. This enables more sparsity in the constrained weights, as any $u_l(\boldsymbol{x})$ can take values of exactly 0. Both methods rely on a "temperature parameter", $\mathcal{T}$, which can be used to control the shape of the projected surface.

5. Applications

This section outlines two different examples of model mixing using the RPBART method for Bayesian Model Mixing (RPBART-BMM). Section 5.1 demonstrates the methodology on a toy simulation example, which combines $K = 2$ simulators, each with two inputs. Both simulators in Section 5.1 provide a high-fidelity approximation of the underlying system in one subregion of the domain and is less accurate across the other regions. Similar patterns are often observed in GCMs. Section 5.2 demonstrates our methodology on a real-data application with eight GCMs that model the average monthly surface temperature. Additional examples of RPBART and RPBART-BMM can be found in Section 7.6 of the supplementary material.

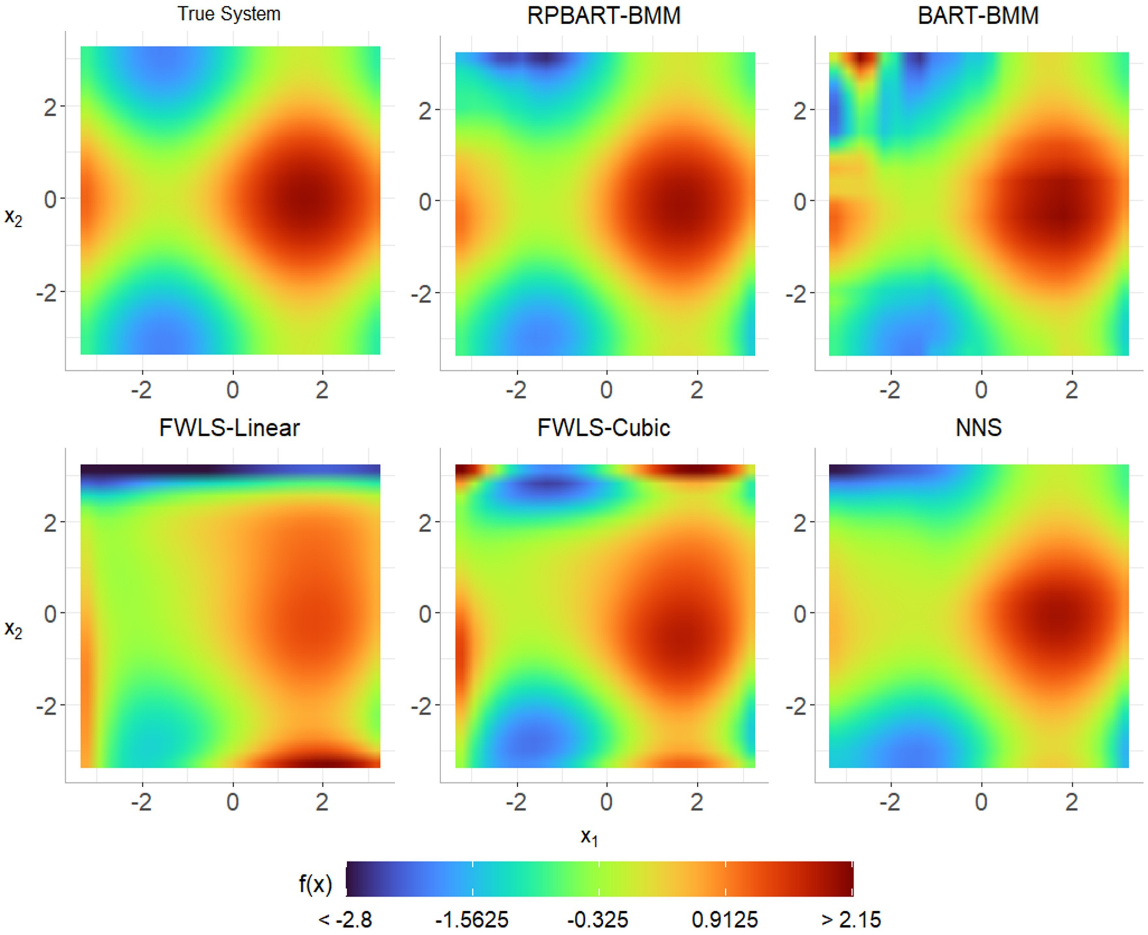


Figure 6: (Top row) The true system, $f_\dagger(\boldsymbol{x})$, (left) vs. the mean predictions, $\hat{f}_\dagger(\boldsymbol{x})$, using the RPBART-BMM (center) or BART-BMM (right). (Bottom row) The mean predictions from Feature Weighted Linear Stacking (FWLS) with a linear basis (left), FWLS with a cubic basis (center), and Neural Network Stacking (NNS) (right).

5.1 A Toy Numerical Experiment

Consider a 2-dimensional example that mixes two simulators in the domain of $[-\pi, \pi] \times [-\pi, \pi]$. Figure 5 illustrates the true underlying function (right) and the two mean predictions from the simulators (center and left). The first simulator (center) provides a high-fidelity approximation of the system in the domain of $[0, \pi] \times [0, \pi]$. Meanwhile, the second simulator (left) provides a high-fidelity approximation when $x_2 < 0$. Neither simulator accurately predicts the true system in the top left corner of the domain. In particular, both simulators take values that are noticeably different than the underlying system.

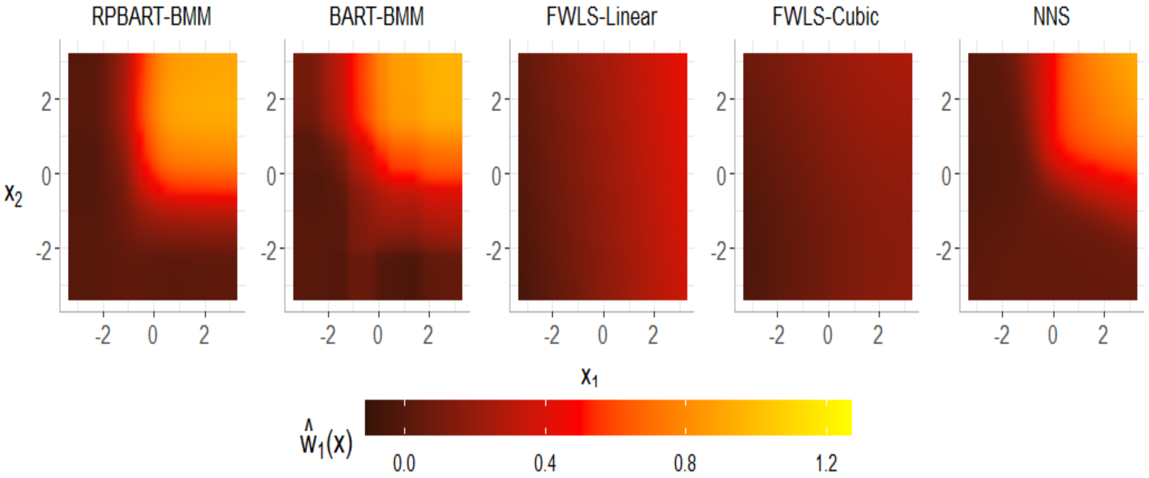


Figure 7: The mean predictions of the weight functions, $\hat{w}_1(\boldsymbol{x})$ using RPBART-BMM, BART-BMM, FWLS-Linear, FWLS-Cubic, and NNS.

A sample of 80 observations were independently generated from the underlying system with mean $f_\dagger(\boldsymbol{x})$ and standard deviation 0.1. The BART-BMM and RPBART-BMM models were fit using this data and the final predictions are shown in Figure 6.

The root mean squared prediction errors are 0.14 and 0.357 for RPBART-BMM and BART-BMM, respectively. The clear distinction between the two mean predictions lies within the top left corner of the domain, as RPBART-BMM does not overfit to the training data.

Additionally, we apply Feature Weighted Linear Stacking (FWLS) (Sill et al., 2009) and Neural Network Stacking (NNS) (Coscrato et al., 2020). The FWLS weight functions are modeled using pre-specified basis functions in terms of x_1 and x_2. Figure 6 displays the mean predictions for FWLS using a linear basis (FWLS-Linear, bottom row, left) and FWLS using a cubic basis (FWLS-Cubic, bottom row, center). Both FWLS predictions struggle to capture the distinct features of the system, particularly around the edges of the domain. In both cases, we observe how pre-specified basis functions may be insufficient for recovering the underlying system across the entire domain. The neural network stacking model (3 hidden layers with 100 nodes per layer) performs better than the linear model, although it also struggles to capture the function behavior around the domain.

The mean weight functions for the first simulator are shown in Figure 7, while the mean weight functions for the second simulator are displayed in Figure 8. Overall, the underlying inference is very similar across the two BART-based methods and NNS. The first simulator receives weight near 1 in the top right corner of the domain where it is the best approximator, the second simulator receives weight near 1 in the bottom half of the domain, and a useful combination of the simulators is determined in the top left corner. Meanwhile, inference using the two FWLS methods is less clear. FWLS-Linear is not flexible enough to produce meaningful inference, as the weights exhibit subtle changes over the domain and neither

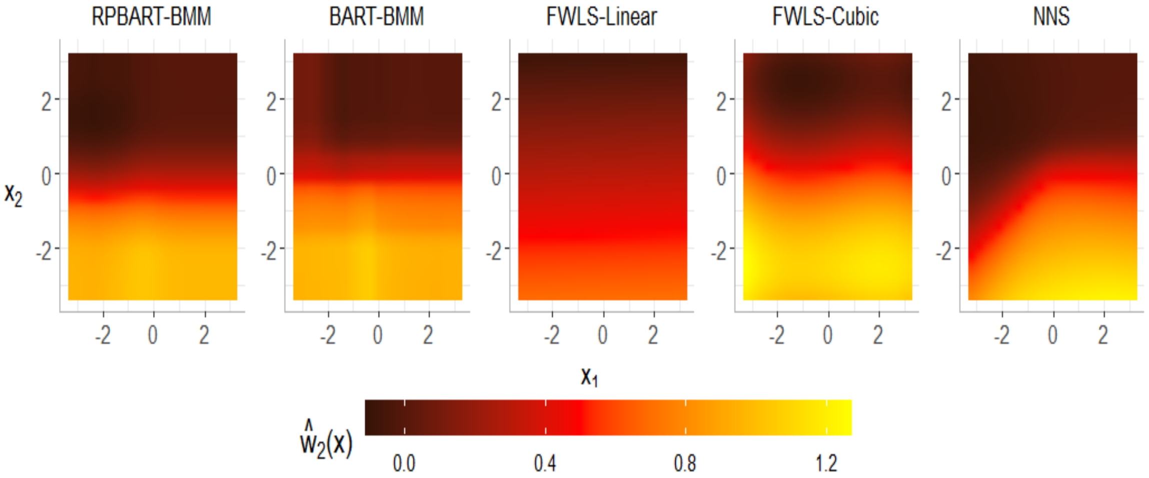


Figure 8: (Top) The mean predictions of the weight functions, $\hat{w}_2(\boldsymbol{x})$ using RPBART-BMM, BART-BMM, FWLS-Linear, FWLS-Cubic, and NNS.

model is heavily favored over the other. The added complexity of the cubic basis allows FWLS-Cubic to yield more reliable inference, particularly with the second weight function.

5.2 Application to Climate Data Integration

In this application, we mix multiple GCMs which model the monthly average two-meter surface temperature (T2M) across the world for April, August, and December 2014. These three time periods are chosen to capture how the GCM performance varies across different months. We combine the output of eight different simulators, each with varying fidelity across the input space and spatial resolution.

We compare the RPBART-BMM approach to Feature-Weighted Linear Stacking and Neural Network Stacking. We allow all three methods to model the weights as functions of latitude, longitude, elevation, and month. Each mixing method is trained using 45,000 observations, where 15,000 are taken from each of the three time periods. The predictions for April, August, and December 2014 are generated over a grid of 259,200 latitude and longitude pairs for each time period.

We downloaded the GCM data from the Coupled Model Intercomparison Project (CMIP6) (Eyring et al., 2016), a data product which includes outputs from a wide range of simulators used to study various climate features. Each GCM may be constructed on a different set of external forcing or socioeconomic factors, hence the fidelity of each climate model is likely different across the globe. Regardless of these factors, each GCM outputs the T2M on a longitude and latitude grid although the grid resolutions can be different. We denote the output from each GCM at a given input $\boldsymbol{x}$ as $f_l(\boldsymbol{x})$ where $l = 1, \ldots, K$.

In climate applications, reanalysis data is often used for observational data. We obtain the European Centre for Medium-Range Forecasts reanalysis (ERA5) data, which combines observed surface temperatures with results from a weather forecasting model to produce

Model	April 2014	August 2014	December 2014
RPBART-BMM	**0.827**	**0.864**	**0.882**
NNS	0.993	1.032	1.088
FWLS	2.393	2.345	2.142

Table 1: The root mean square error for each mixing approach (BART-BMM, NNS, and FWLS) stratified by month. Each RMSE is computed by evaluating the output of the mixed-prediction or a given GCM over a dense grid of 259,200 inputs and comparing to the observed ERA5 data. Smallest RMSEs for each month are denoted in bold.

measurements of T2M across a dense grid of 0.25° longitude by 0.25° latitude. We denote the ith reanalysis data point as $Y(\boldsymbol{x}_i)$ where $\boldsymbol{x}_i$ denotes the ith latitude, longitude, elevation, and month for $i = 1, \ldots, 45000$. The ERA5 data and elevation data was downloaded from the Copernicus Climate Data Store.

A common assumption in model mixing is that the simulators are evaluated across the same grid of inputs as the observed response data. To map the data onto the same grid, we apply the bilinear interpolation to obtain an inexpensive emulator, $\hat{f}_1(\boldsymbol{x}), \ldots, \hat{f}_K(\boldsymbol{x})$, for each GCM (National Center for Atmospheric Research Staff , Eds).

Table 1 displays the root mean squared errors for each of the three time periods. The RPBART-BMM model performs the best across each month. We also note each of the three mixing methods outperform the individual GCMs as shown in Section 7.6.3 of the Supplementary Material. Figure 9 displays the ERA5 data (left) and the mean predictions from the three approaches across four different regions in April 2014. In general, RPBART-BMM and NNS result in very similar predictions that capture the granular features in the data. Some of the granularity is lost in FWLS due to the less flexible form of the weight functions defined by a linear basis. Only subtle differences exist between RPBART-BMM and NNS. Also, RPBART-BMM better leverages elevation to preserve the fine details in the data, as seen across the Suntar-Khayata Mountains and the Tibetan Plateau predictions.

In addition to the improved mean prediction, we are also interested in identifying the subregions where each simulator is favored in the ensemble. In most cases, we have observed that weight values closer to 1 typically indicates more accurate GCMs, while weights near 0 typically indicate less accurate GCMs. Figure 10 displays the posterior mean weight functions for the eight simulators in the Northwestern hemisphere in April 2014. The posterior mean weights in Figure 10 suggest CNRM, CESM2, and MIROC are more active in the mixed-prediction across the western part of the United States and Canada, as these three GCMs receive higher weight relative to the other GCMs. In the same region, we see BCC and CanESM5 receive negative weights, which are possibly due to multicollinearity and can be investigated using posterior weight projections as described in Section 4.2.

To better identify a subset of GCMs that are active in a specific region, we consider projecting the samples of the posterior weight functions onto the simplex using a penalized L_2 projection (the sparsegen-linear projection) (Laha et al., 2018). This imposes a sum-to-one and non-negativity constraint on $\boldsymbol{u}(\boldsymbol{x})$. For this example $\mathcal{T} = 0.137$ by minimizing the sum of squared differences between the predicted mean with unconstrained weights and the

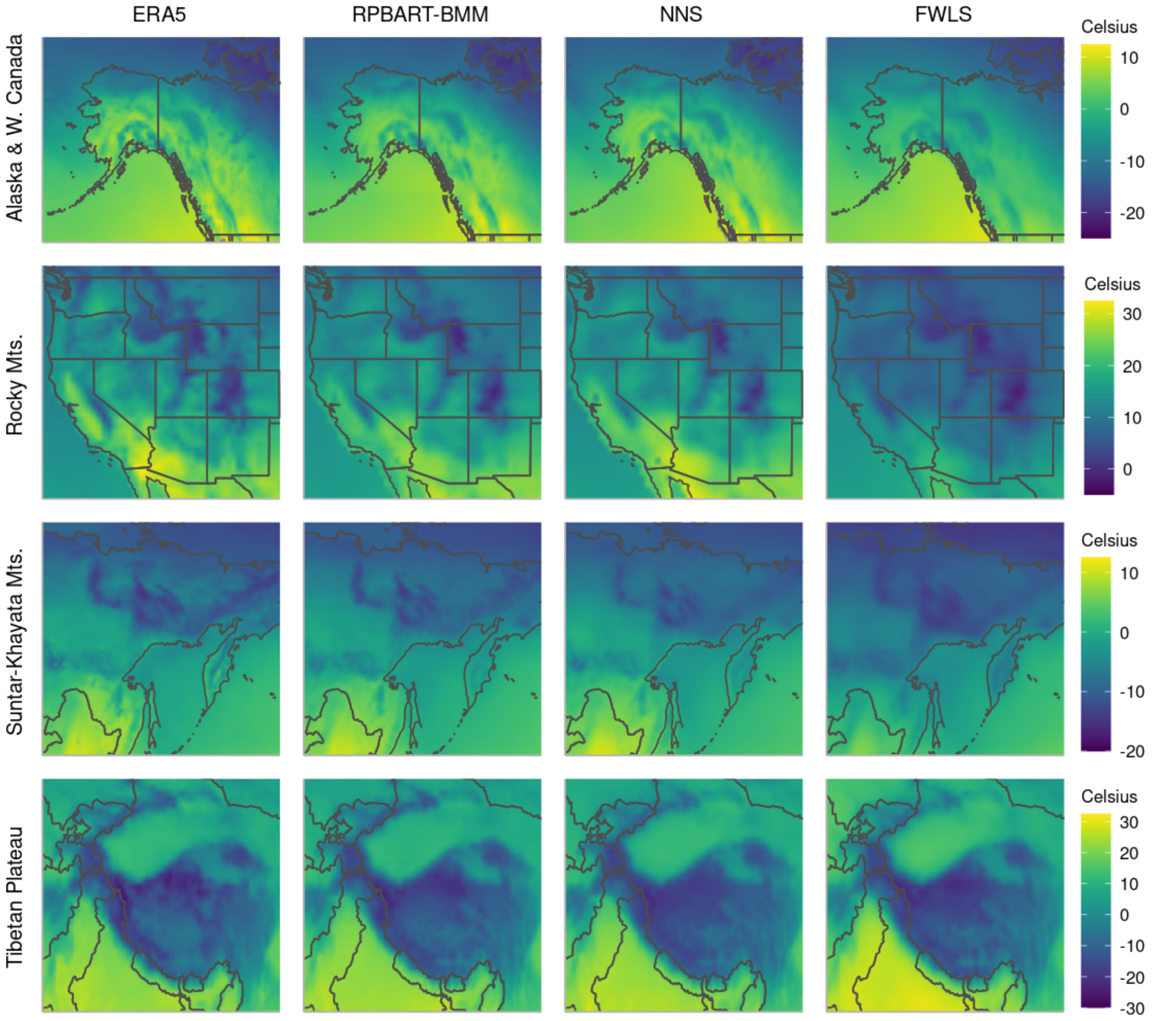


Figure 9: The ERA5 data (first column) vs the mean predictions from RPBART-BMM (second column), NNS (third column), or FWLS (fourth column). Each row highlights different regions of the world in April 2014.

predicted mean with the constrained weights (this difference is defined as the discrepancy function), $\sum_{i=1}^{N_v} \left(\delta(\boldsymbol{x}_i^v)\right)^2$, at $N_v = 5000$ validation points.

The posterior mean predictions from the constrained mixture of GCMs and the posterior mean discrepancy are shown in Figure 11. Overall, the constrained mixed-prediction recovers the major features of the underlying temperature patterns, however some fine details are lost due to the simplex constraint. This, at times, can be indicative of response features that are not accounted for by any of the GCMs in the model set. Typically, we lose some of the granularity in the areas with significant elevation changes, such as the Rocky Mountains (second row). The constrained weights are less flexible and unable to properly account for

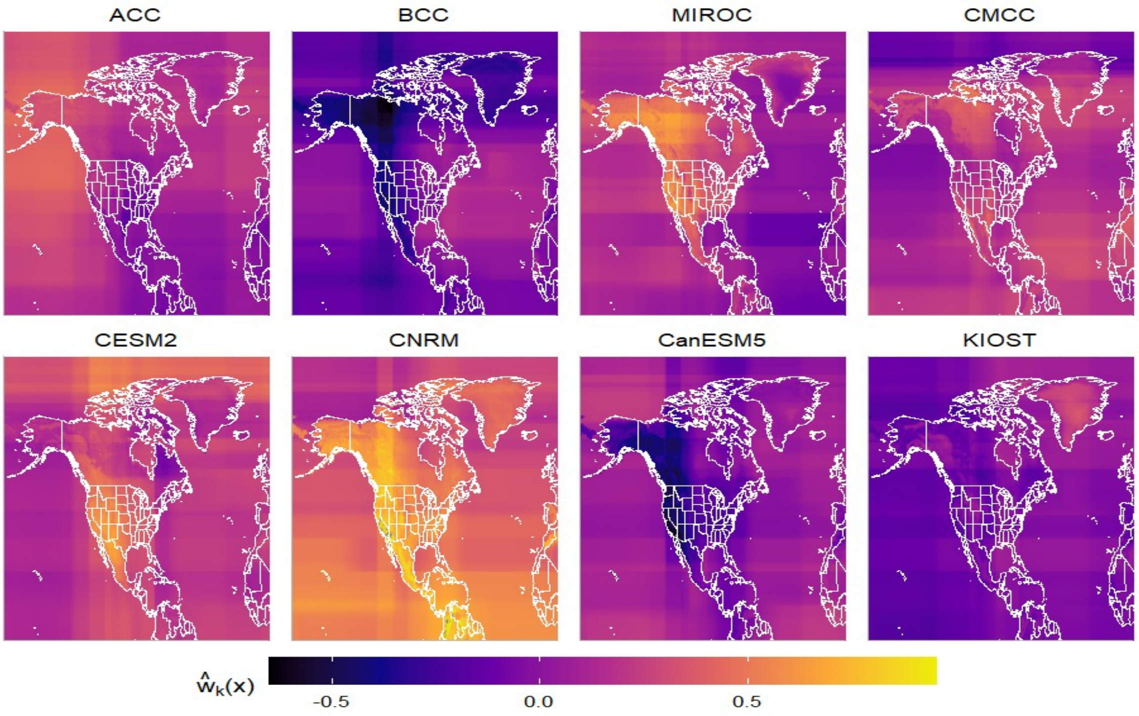


Figure 10: The posterior mean of the RPBART-based weight functions for each of the eight simulators across the northwestern hemisphere in April 2014.

the discrepancy within the GCMs. The remaining variability that is unaccounted for by the constrained mixed-prediction is then attributed to the additive discrepancy.

Figure 12 displays the projected weight functions for April 2014. Similar conclusions can be made as in the unconstrained weights from Figure 10, however in the constrained model, we can clearly isolate the GCMs that contribute to the prediction in Figure 11. For example, we see CNRM, CESM2, and MIROC receive relatively high weight in the western part of the United States and Canada. Coupled with the discrepancy in Figure 11, these three GCMs are sufficient for recovering the system in Alaska and Western Canada (low $\delta(\boldsymbol{x})$), but are insufficient for recovering the system in the Rocky Mountains (higher $\delta(\boldsymbol{x})$).

Finally, we can connect the unconstrained and constrained models in terms of how each identifies model discrepancy. In the unconstrained model, the posterior sum of the weight functions, $w_{sum}(\boldsymbol{x}) = \sum_{l=1}^{K} w_l(\boldsymbol{x})$, is one metric that can be used to help identify model discrepancy. In our applications (both climate and nuclear physics), we have observed that the posterior distribution of $w_{sum}(\boldsymbol{x})$ deviates away from 1 in areas where the GCMs do not accurately model the underlying system. This phenomenon has also been observed in earlier work with constant weights (Breiman, 1996; Le and Clarke, 2017). The top panel of Figure 13 displays the posterior mean of $w_{sum}(\boldsymbol{x})$ over selected regions in April 2014. The bottom row of Figure 13 displays the posterior mean of $\delta(\boldsymbol{x})$ from the constrained model in April 2014. Subregions where $w_{sum}(\boldsymbol{x})$ deviates away from 1 (purple or green) align well

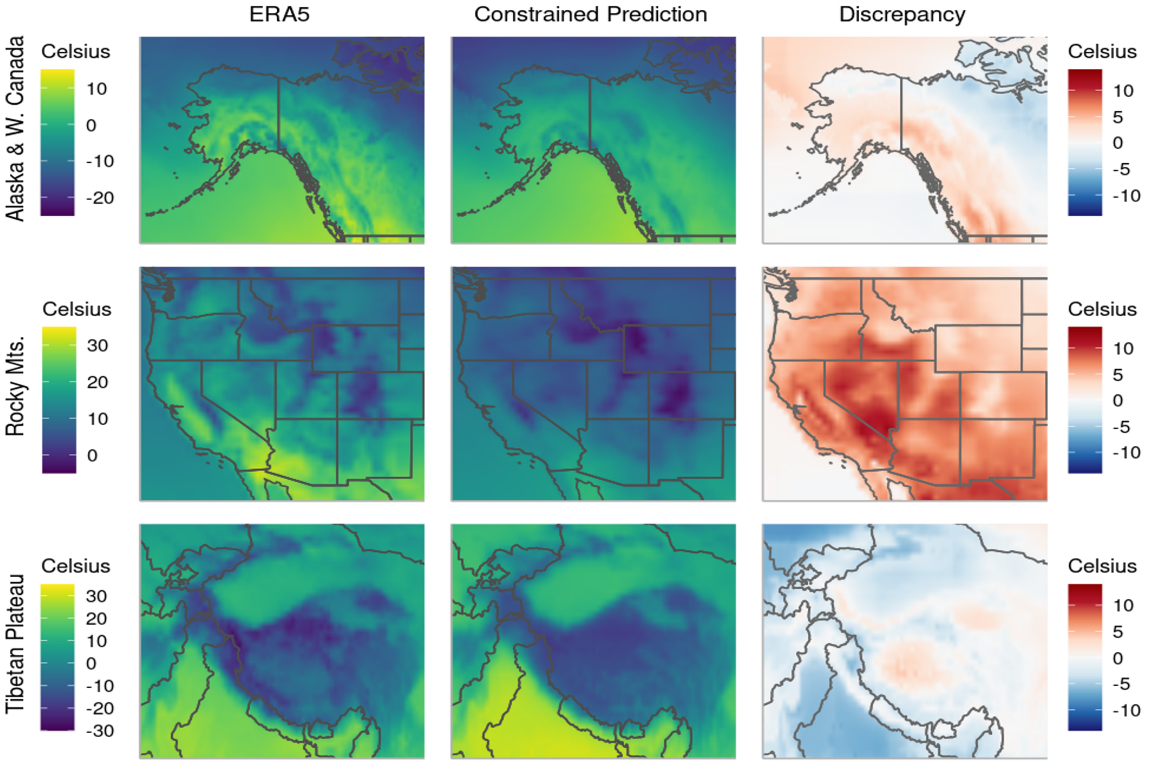


Figure 11: The ERA5 data (left) vs. the simplex constrained mixture of simulators (center) and the additive discrepancy (right) across three subregions (rows) in April 2014.

with the areas where $\delta(\boldsymbol{x})$ deviates away from 0 (orange or blue). Thus, the post-processing approach used to estimate $\delta(\boldsymbol{x})$ preserves the information in the posterior of $w_{sum}(\boldsymbol{x})$.

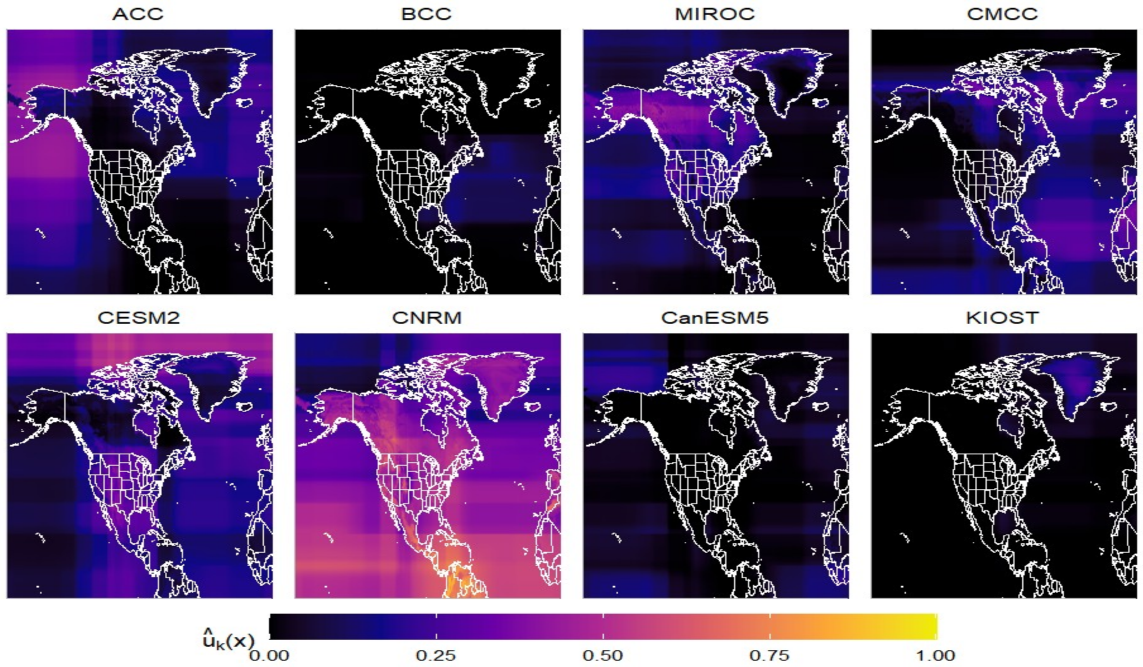


Figure 12: The posterior mean of the constrained RPBART-based weight functions for each of the eight simulators across the northwestern hemisphere in April 2014.

6. Discussion

This research proposes a random path model as a novel approach to introduce continuity into the sum-of-trees model. We then extend the original mean mixing framework presented by Yannotty et al. (2024) to combine the outputs from a general collection of GCMs while also modeling the weight functions as continuous functions. This methodology has been successfully applied to GCMs, which model the monthly average surface temperature.

This model mixing approach depends on information from the observational data and all GCM outputs. Our method explicitly assumes one can evaluate each GCM, or an emulator for the GCM, at the n inputs associated with the observational data. We construct each emulator using bilinear interpolation with respect to the latitude and longitude grid associated with the monthly temperature output. This results in a simple, yet lower resolution emulator for each GCM. Since the emulators maintain a lower resolution, the weight functions in the RPBART-BMM model are left to account for the granularity in the observed data. Thus, lower resolution emulators require more complex weight functions. For RPBART-BMM, the required complexity corresponds to deep trees, less smoothing, and larger m.

Alternatively, one could replace the bilinear interpolation approach with a more complex emulator, such as another RPBART model, neural networks, or Gaussian processes (Watson-Parris, 2021; Kasim et al., 2021). Such emulators could depend on more features

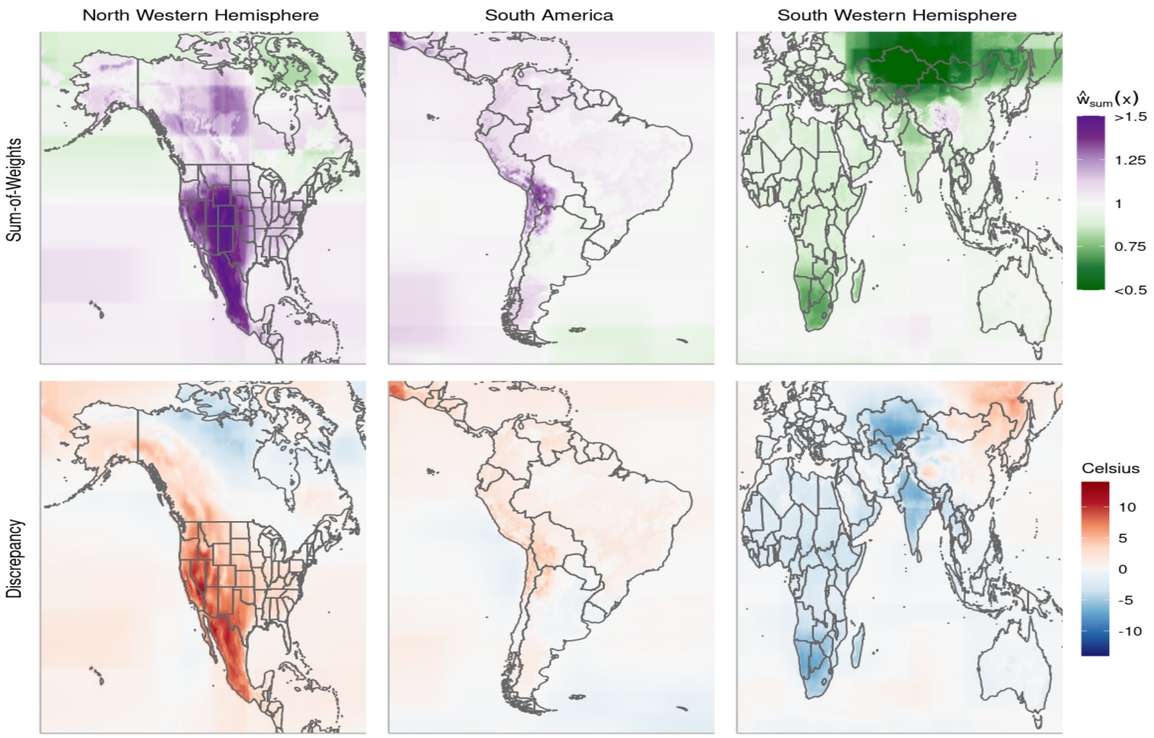


Figure 13: The sum-of-weights from the unconstrained RPBART-BMM model (top) and the additive discrepancy from the constrained weight model (bottom). Each column corresponds to a different region in April 2014.

other than longitude and latitude and thus result in higher fidelity interpolations of the GCM output. Higher resolution emulators would likely explain more granularity in the data and thus result in less complex weight functions (i.e. shallower trees, more smoothing, and smaller m).

Existing climate ensembles tend to focus on the temporal distribution of the simulator output and combine the GCMs in a pointwise manner for each latitude and longitude pair (Vrac et al., 2024; Harris et al., 2023). Though a temporal component can be considered as an input to the weight functions, we mainly focus on learning the spatial distribution of the weights to help identify subregions where each GCM is influential. Future work will further explore other ways to incorporate the temporal distribution of the GCMs, which could allow for more appropriate long term model-mixed predictions of future temperature.

7. Acknowledgements

The work of JCY was supported in part by the National Science Foundation under Agreement OAC-2004601. The work of MTP was supported in part by the National Science Foundation under Agreements DMS-1916231, DMS-1564395, OAC-2004601. The work of

TJS was supported in part by the National Science Foundation under Agreement DMS-1564395 (The Ohio State University). The work of BL was supported in part by the National Science Foundation under Agreement DMS-2503494.

References

Leo Breiman. Stacked regressions. *Machine learning*, 24(1):49–64, 1996.

H. Chipman, E. George, and R. McCulloch. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian CART model search. *Journal of the American Statistical Association*, 93(443):935–948, 1998.

Merlise Clyde and Edwin S Iversen. Bayesian model averaging in the M-open framework. *Bayesian theory and applications*, pages 484–498, 2013.

Victor Coscrato, Marco Henrique de Almeida Inácio, and Rafael Izbicki. The nn-stacking: Feature weighted linear stacking through neural networks. *Neurocomputing*, 399:141–152, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2020.02.073.

Noel Cressie. *Statistics for spatial data*. John Wiley & Sons, 2015.

David GT Denison, Bani K Mallick, and Adrian FM Smith. A bayesian cart algorithm. *Biometrika*, 85(2):363–377, 1998.

Veronika Eyring, Sandrine Bony, Gerald A Meehl, Catherine A Senior, Bjorn Stevens, Ronald J Stouffer, and Karl E Taylor. Overview of the coupled model intercomparison project phase 6 (cmip6) experimental design and organization. *Geoscientific Model Development*, 9(5):1937–1958, 2016.

Filippo Giorgi and Linda O Mearns. Calculation of average, uncertainty range, and reliability of regional climate changes from aogcm simulations via the "reliability ensemble averaging"(rea) method. *Journal of climate*, 15(10):1141–1158, 2002.

Emmanuel Gringarten and Clayton V Deutsch. Teacher's aide variogram interpretation and modeling. *Mathematical Geology*, 33:507–534, 2001.

Trevor Harris, Bo Li, and Ryan Sriver. Multimodel ensemble analysis with neural network gaussian processes. *The Annals of Applied Statistics*, 17(4):3403–3425, 2023.

Trevor Hastie and Robert Tibshirani. Bayesian backfitting (with comments and a rejoinder by the authors. *Statistical Science*, 15(3):196–223, 2000.

Ozan Irsoy, Olcay Taner Yıldız, and Ethem Alpaydın. Soft decision trees. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 1819–1822. IEEE, 2012.

Muhammad Firmansyah Kasim, Duncan Watson-Parris, Lucia Deaconu, Sophy Oliver, P Hatfield, Dustin H Froula, Gianluca Gregori, Matt Jarvis, Samar Khatiwala, Jun Korenaga, et al. Building high accuracy emulators for scientific simulations with deep neural architecture search. *Machine Learning: Science and Technology*, 3(1):015013, 2021.

Vojtech Kejzlar, Léo Neufcourt, and Witold Nazarewicz. Local bayesian dirichlet mixing of imperfect models. *Scientific Reports*, 13(1):19600, 2023.

Reto Knutti, Jan Sedláček, Benjamin M Sanderson, Ruth Lorenz, Erich M Fischer, and Veronika Eyring. A climate model projection weighting scheme accounting for performance and interdependence. *Geophysical Research Letters*, 44(4):1909–1918, 2017.

Weiwei Kong, Walid Krichene, Nicolas Mayoraz, Steffen Rendle, and Li Zhang. Rankmax: An adaptive projection alternative to the softmax function. *Advances in Neural Information Processing Systems*, 33:633–643, 2020.

Anirban Laha, Saneem Ahmed Chemmengath, Priyanka Agrawal, Mitesh Khapra, Karthik Sankaranarayanan, and Harish G Ramaswamy. On controllable sparse alternatives to softmax. *Advances in neural information processing systems*, 31, 2018.

Tri Le and Bertrand Clarke. A Bayes interpretation of stacking for M-complete and M-open settings. *Bayesian Analysis*, 12(3):807–829, 2017.

Lizhen Lin and David B Dunson. Bayesian monotone regression using gaussian process projection. *Biometrika*, 101(2):303–317, 2014.

Antonio R Linero. A review of tree-based bayesian methods. *Communications for Statistical Applications and Methods*, 24(6), 2017.

Antonio R Linero and Yun Yang. Bayesian regression tree ensembles that adapt to smoothness and sparsity. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 80(5):1087–1110, 2018.

Fei Liu, MJ Bayarri, JO Berger, et al. Modularization in bayesian analysis, with emphasis on analysis of computer models. *Bayesian Analysis*, 4(1):119–150, 2009.

Georges Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963.

National Center for Atmospheric Research Staff (Eds). The climate data guide: Regridding overview. `https://climatedataguide.ucar.edu/climate-tools/regridding-overview`, 2014. Accessed: 2024-03-11.

DR Phillips, RJ Furnstahl, U Heinz, T Maiti, W Nazarewicz, FM Nunes, M Plumlee, MT Pratola, S Pratt, FG Viens, et al. Get on the BAND wagon: a Bayesian framework for quantifying model uncertainties in nuclear dynamics. *Journal of Physics G: Nuclear and Particle Physics*, 48(7):072001, 2021.

Matthew T Pratola. Efficient Metropolis–Hastings proposal mechanisms for Bayesian regression tree models. *Bayesian analysis*, 11(3):885–911, 2016.

A. Raftery, D. Madigan, and J. Hoeting. Bayesian model averaging for linear regression models. *Journal of the American Statistical Association*, 92(437):179–191, 1997.

Philip G Sansom, David B Stephenson, and Thomas J Bracegirdle. On constraining projections of future climate using observations and simulations from multiple climate models. *Journal of the American Statistical Association*, 116(534):546–557, 2021.

Thomas J Santner, Brian J Williams, and William I Notz. *The Design and Analysis of Computer Experiments*. Springer, 2nd edition, 2018.

A. C. Semposki, R. J. Furnstahl, and D. R. Phillips. Interpolating between small- and large-g expansions using bayesian model mixing. *Phys. Rev. C*, 106:044002, Oct 2022. doi: 10.1103/PhysRevC.106.044002.

Deborshee Sen, Sayan Patra, and David Dunson. Constrained inference through posterior projections. *arXiv preprint arXiv:1812.05741*, 2018.

Joseph Sill, Gábor Takács, Lester Mackey, and David Lin. Feature-weighted linear stacking. *arXiv preprint arXiv:0911.0460*, 2009.

Claudia Tebaldi and Reto Knutti. The use of the multi-model ensemble in probabilistic climate projections. *Philosophical transactions of the royal society A: mathematical, physical and engineering sciences*, 365(1857):2053–2075, 2007.

Mathieu Vrac, Denis Allard, Grégoire Mariéthoz, Soulivanh Thao, and Lucas Schmutz. Distribution-based pooling for combination and multi-model bias correction of climate simulations. *EGUsphere*, 2024:1–41, 2024.

Duncan Watson-Parris. Machine learning for weather and climate are worlds apart. *Philosophical Transactions of the Royal Society A*, 379(2194):20200098, 2021.

John C Yannotty. *Bayesian Additive Regression Tree Methodology for Multi-Simulator Computer Experiments*. PhD thesis, The Ohio State University, 2024.

John C. Yannotty, Thomas J. Santner, Richard J. Furnstahl, and Matthew T. Pratola. Model mixing using bayesian additive regression trees. *Technometrics*, 66(2):196–207, 2024. doi: 10.1080/00401706.2023.2257765.

Y. Yao, A. Vehtari, D. Simpson, and A. Gelman. Using stacking to average Bayesian predictive distributions. *Bayesian Analysis*, 13(3):917–1007, 2018.

Y. Yao, G. Pirš, A. Vehtari, and A. Gelman. Bayesian hierarchical stacking: Some models are (somewhere) useful. *Bayesian Analysis*, 1(1):1–29, 2021.

Supplement

7.1 The Conditional Covariance Model

We can better understand the parameters in the RPBART model by studying their effects on the covariance of the response for two inputs, $\boldsymbol{x}$ and $\boldsymbol{x}'$. Conditional on the set of trees, $T_1, \ldots, T_m$, bandwidth parameters $\gamma_1, \ldots, \gamma_m$, and error variance σ^2, the prior covariance between $Y(\boldsymbol{x})$ and $Y(\boldsymbol{x}')$ is given by Theorem 7.1.

Theorem 7.1 (Conditional Covariance) *Assume the m sets of terminal node parameters,*
$M_1, \ldots, M_m$, and random path assignments, $Z_1, \ldots, Z_m$ are mutually independent conditional on the set of trees and bandwidth parameters. Further assume that the random path assignments for an observation $\boldsymbol{x}$ are conditionally independent of those for another input $\boldsymbol{x}'$. Then, conditional on the set of trees, $T_1, \ldots, T_m$, bandwidth parameters $\gamma_1, \ldots, \gamma_m$, and error variance σ^2 the prior covariance between $Y(\boldsymbol{x})$ and $Y(\boldsymbol{x}')$ when $\boldsymbol{x} \neq \boldsymbol{x}'$ is given by

$$Cov\big(Y(\boldsymbol{x}),\ Y(\boldsymbol{x}') \mid \Theta\big) = m\tau^2 \sum_{j=1}^{m} \frac{1}{m} \sum_{b=1}^{B_j} \phi_{bj}(\boldsymbol{x}; T_j, \gamma_j)\phi_{bj}(\boldsymbol{x}'; T_j, \gamma_j), \tag{13}$$

where $\Theta = \big\{\{T_j, \gamma_j\}_{j=1}^m, \sigma^2\big\}$. The conditional variance is given by

$$Var\big(Y(\boldsymbol{x}) \mid \Theta\big) = m\tau^2 + \sigma^2. \tag{14}$$

Proof Fix the set of trees, bandwidth parameters, and σ^2 and let $\Theta = \Big\{\{T_j, \gamma_j\}_{j=1}^m, \sigma^2\Big\}$. By definition, the conditional covariance between $Y(\boldsymbol{x})$ and $Y(\boldsymbol{x}')$ is given by

$$\begin{aligned}\text{Cov}\Big(Y(\boldsymbol{x}),\ Y(\boldsymbol{x}') \mid \Theta\Big) &= \text{Cov}\Big(\sum_{j=1}^{m} g(\boldsymbol{x}; T_j, M_j, Z_j) + \epsilon(\boldsymbol{x}), \\ &\qquad\qquad \sum_{k=1}^{m} g(\boldsymbol{x}'; T_k, M_k, Z_k) + \epsilon(\boldsymbol{x}') \mid \Theta\Big) \\ &= \sum_{j=1}^{m}\sum_{k=1}^{m} \text{Cov}\big(g(\boldsymbol{x}; T_j, M_j, Z_j),\ g(\boldsymbol{x}'; T_k, M_k, Z_k) \mid \Theta\big).\end{aligned}$$

where $g(\boldsymbol{x}; T_j, M_j, Z_j) = \sum_{b=1}^{B_j} \mu_{bj} z_{bj}(\boldsymbol{x})$ and $\boldsymbol{x} \neq \boldsymbol{x}'$. Due to the conditional independence assumption across the m trees and associated parameters, the covariance simplifies as

$$\text{Cov}\Big(Y(\boldsymbol{x}),\ Y(\boldsymbol{x}') \mid \Theta\Big) = \sum_{j=1}^{m} \text{Cov}\Big(g(\boldsymbol{x}; T_j, M_j, Z_j),\ g(\boldsymbol{x}'; T_j, M_j, Z_j) \mid \Theta\Big), \tag{15}$$

since $\mathrm{Cov}\big(g(\boldsymbol{x}; T_j, M_j, Z_j),\ g(\boldsymbol{x}'; T_k, M_k, Z_k) \mid \Theta\big) = 0$ when $j \neq k$. Finally, we can consider the covariance function within the jth tree, which simplifies as follows

$$\begin{aligned}\mathrm{Cov}\Big(g(\boldsymbol{x}; T_j, M_j, Z_j),\ g(\boldsymbol{x}'; T_j, M_j, Z_j) \mid \Theta\Big) &= \mathrm{Cov}\Big(\sum_{b=1}^{B_j} \mu_{bj} z_{bj}(\boldsymbol{x}),\ \sum_{d=1}^{B_j} \mu_{dj} z_{dj}(\boldsymbol{x}') \mid \Theta\Big)\\ &= \sum_{b=1}^{B_j}\sum_{d=1}^{B_j} \mathrm{Cov}\Big(\mu_{bj} z_{bj}(\boldsymbol{x}),\ \mu_{dj} z_{dj}(\boldsymbol{x}') \mid \Theta\Big).\end{aligned}$$

Once again, conditional independence between the terminal node parameters and random path assignments implies $\mathrm{Cov}\big(\mu_{bj} z_{bj}(\boldsymbol{x}),\ \mu_{dj} z_{dj}(\boldsymbol{x}')\big) = 0$ when $b \neq d$. However, when $b = d$ the covariance within the jth tree is defined by

$$\begin{aligned}\sum_{b=1}^{B_j} \mathrm{Cov}\big(\mu_{bj} z_{bj}(\boldsymbol{x}),\ \mu_{bj} z_{bj}(\boldsymbol{x}') \mid \Theta\big) &= \sum_{b=1}^{B_j} E\big[\mu_{bj}^2 z_{bj}(\boldsymbol{x}) z_{bj}(\boldsymbol{x}') \mid \Theta\big]\\ &\quad - E\big[\mu_{bj} z_{bj}(\boldsymbol{x}) \mid \Theta\big] E\big[\mu_{bj} z_{bj}(\boldsymbol{x}') \mid \Theta\big]\\ &= \tau^2 \sum_{b=1}^{B_j} \phi_{bj}(\boldsymbol{x}; T_j, \gamma_j)\ \phi_{bj}(\boldsymbol{x}'; T_j, \gamma_j),\end{aligned}$$

where $E\big[\mu_{bj} z_{bj}(\boldsymbol{x})\big] = 0$ because each terminal node parameter has mean zero and we assume conditional independence between each μ_{bj} and $z_{bj}(\boldsymbol{x})$. Returning to (15), the covariance between $Y(\boldsymbol{x})$ and $Y(\boldsymbol{x}')$ when $\boldsymbol{x} \neq \boldsymbol{x}'$ is then defined as

$$\mathrm{Cov}\big(Y(\boldsymbol{x}),\ Y(\boldsymbol{x}') \mid \Theta\big) = m\tau^2 \sum_{j=1}^{m} \frac{1}{m} \sum_{b=1}^{B_j} \phi_{bj}(\boldsymbol{x}; T_j, \gamma_j) \phi_{bj}(\boldsymbol{x}'; T_j, \gamma_j).$$

An expression for the conditional variance of $Y(\boldsymbol{x})$ is also given by

$$\begin{aligned}\mathrm{Var}\big(Y(\boldsymbol{x}) \mid \Theta\big) &= \mathrm{Var}\big(\sum_{j=1}^{m} g(\boldsymbol{x}; T_j, M_j, Z_j) + \epsilon(\boldsymbol{x}) \mid \Theta\big)\\ &= \sum_{j=1}^{m} \mathrm{Var}\big(g(\boldsymbol{x}; T_j, M_j, Z_j) \mid \Theta\big) + \mathrm{Var}\big(\epsilon(\boldsymbol{x}) \mid \Theta\big). \qquad (16)\end{aligned}$$

Due to the conditional independence between parameters across trees, we can once again focus on the variance within each tree separately. The conditional variance for the output of the jth tree model, is given by

$$\begin{aligned}
\mathrm{Var}\Big(g(\boldsymbol{x};T_j,M_j,Z_j) \mid \Theta\Big) &= \mathrm{Var}\Big(\sum_{b=1}^{B_j} \mu_{bj} z_{bj}(\boldsymbol{x}) \mid \Theta\Big) \\
&= E\Big[\Big(\sum_{b=1}^{B_j} \mu_{bj} z_{bj}(\boldsymbol{x})\Big)^2 \mid \Theta\Big] - E\Big[\sum_{b=1}^{B_j} \mu_{bj} z_{bj}(\boldsymbol{x}) \mid \Theta\Big]^2 \\
&= \sum_{b=1}^{B_j}\sum_{d=1}^{B_j} E\Big[\mu_{bj}\mu_{dj} z_{bj}(\boldsymbol{x}) z_{dj}(\boldsymbol{x}) \mid \Theta\Big].
\end{aligned}$$

By assumption, $z_{bj}(\boldsymbol{x}) \in \{0,1\}$ and $\sum_{b=1}^{B_j} z_{bj}(\boldsymbol{x}) = 1$. Thus, if $b \neq d$, then $z_{bj}(\boldsymbol{x})z_{dj}(\boldsymbol{x}) = 0$ with probability 1. The conditional variance then simplifies as

$$\begin{aligned}
\mathrm{Var}\big(g(\boldsymbol{x};T_j,M_j,Z_j) \mid \Theta\big) &= \sum_{b=1}^{B_j} E\big[\mu_{bj}^2 z_{bj}^2(\boldsymbol{x}) \mid \Theta\big] \\
&= \sum_{b=1}^{B_j} E\big[\mu_{bj}^2 z_{bj}(\boldsymbol{x}) \mid \Theta\big] \\
&= \sum_{b=1}^{B_j} \tau^2 \phi_{bj}(\boldsymbol{x};T_j,\gamma_j) \\
&= \tau^2,
\end{aligned}$$

where $\sum_{b=1}^{B_j} \phi_{bj}(\boldsymbol{x};T_j,\gamma_j) = 1$. Thus, the conditional variance of the tree output is simply the variance of the terminal node parameters τ^2. The variance of the sum-of-tress is then given by $m\tau^2$. These results are the same as in the original BART model. Returning to (16), the conditional variance of $Y(\boldsymbol{x})$ is given by $\mathrm{Var}\big(Y(\boldsymbol{x}) \mid \Theta\big) = m\tau^2 + \sigma^2$.

■

7.2 Proof of the Semivariogram Formula

Consider the semivariogram of the sum-of-trees model, which depends on the function $\nu(\boldsymbol{x},\boldsymbol{h})$. Due to the constant mean assumption in RPBART, $\nu(\boldsymbol{x},\boldsymbol{h})$ is defined by

$$\begin{aligned}
\nu(\boldsymbol{x},\boldsymbol{h}) &= \frac{1}{2} E^{T,M,Z,\gamma}\Big[\big(Y(\boldsymbol{x}+\boldsymbol{h}) - Y(\boldsymbol{x})\big)^2 \mid \sigma^2\Big] \\
&= \frac{1}{2} E^{T,\gamma}\Big[E^{M,Z}\Big[\big(Y(\boldsymbol{x}+\boldsymbol{h}) - Y(\boldsymbol{x})\big)^2 \mid \Theta\Big]\Big], \qquad (17)
\end{aligned}$$

where $E^{T,\gamma}$ denotes the expectation with respect to the set of trees and bandwidth parameters and $E^{M,Z}$ denotes the conditional expectation with respect to the set of terminal nodes and random path assignments. Moving forward, we will let $\Theta = \big\{\{T_j, \gamma_j\}_{j=1}^m, \sigma^2\big\}$ and treat σ^2 as a fixed value rather than a random variable.

First consider the inner expectation, which will enable us to understand the effect of the priors we assign to each M_j and Z_j in the model. Due to the constant mean assumption, the inner expectation is equivalently expressed as

$$\begin{aligned} E^{M,Z}\Big[\big(Y(\boldsymbol{x}+\boldsymbol{h}) - Y(\boldsymbol{x})\big)^2 \mid \Theta\Big] = {} & \mathrm{Var}\big(Y(\boldsymbol{x}+\boldsymbol{h}) \mid \Theta\big) \\ & + \mathrm{Var}\big(Y(\boldsymbol{x}) \mid \Theta\big) \\ & - 2\,\mathrm{Cov}\big(Y(\boldsymbol{x}+\boldsymbol{h}), Y(\boldsymbol{x}) \mid \Theta\big). \end{aligned} \tag{18}$$

The inner expectation from (18) divided by 2 will be denoted as $\nu(\boldsymbol{x}, \boldsymbol{h}; \Theta)$. Thus, we can focus on $\nu(\boldsymbol{x}, \boldsymbol{h}; \Theta)$, which involves analytically tractable terms, as shown in Theorem 7.1:

$$\nu(\boldsymbol{x}, h; \Theta) = \sigma^2 + m\tau^2\Big(1 - \frac{1}{m}\sum_{j=1}^{m}\sum_{b=1}^{B_j} \phi_{bj}(\boldsymbol{x}+\boldsymbol{h}; T_j, \gamma_j)\phi_{bj}(\boldsymbol{x}; T_j, \gamma_j)\Big). \tag{19}$$

The function $\nu(\boldsymbol{x}, \boldsymbol{h})$ is obtained by computing the outer expectation in (17), which is with respect to the set of m trees and bandwidth parameters. We can then obtain $\nu(\boldsymbol{x}, \boldsymbol{h})$ by marginalizing over the remaining parameters,

$$\begin{aligned} \nu(\boldsymbol{x}, \boldsymbol{h}) &= \sigma^2 + m\tau^2 E\Big[\Big(1 - \frac{1}{m}\sum_{j=1}^{m}\sum_{b=1}^{B_j} \phi_{bj}(\boldsymbol{x}+\boldsymbol{h}; T_j, \gamma_j)\phi_{bj}(\boldsymbol{x}; T_j, \gamma_j)\Big)\Big] \\ &= \sigma^2 + m\tau^2\Big(1 - \frac{1}{m}\sum_{j=1}^{m} E\Big[\sum_{b=1}^{B_j} \phi_{bj}(\boldsymbol{x}+\boldsymbol{h}; T_j, \gamma_j)\phi_{bj}(\boldsymbol{x}; T_j, \gamma_j)\Big]\Big), \end{aligned} \tag{20}$$

where the expectation is with respect to Θ and σ^2 is treated as a fixed constant.

Since the set of trees and bandwidth parameters are sets of i.i.d. random quantities, the expectation in (20) is the same for each $j = 1, \ldots, m$. Finally, we take $\tau = \big(y_{max} - y_{min}\big)/(2k\sqrt{m})$ which implies $m\tau^2$ simplifies as

$$m\tau^2 = \Big(\frac{y_{\max} - y_{\min}}{2k}\Big)^2,$$

where $y_{\max} - y_{\min}$ is the range of the observed data and k is a tuning parameter that controls the flexibility of the sum-of-trees model. Given these two simplifications, $\nu(\boldsymbol{x}, \boldsymbol{h})$ for the RPBART is given by

$$\nu(\boldsymbol{x}, \boldsymbol{h}) = \sigma^2 + \Big(\frac{y_{\max} - y_{\min}}{2k}\Big)^2\Big(1 - E\Big[\sum_{b=1}^{B_1} \phi_{b1}(\boldsymbol{x}+\boldsymbol{h}; \gamma_1, T_1)\phi_{b1}(\boldsymbol{x}; \gamma_1, T_1)\Big]\Big).$$

■

7.3 Applications and Interpretations of the Semivariogram

A semivariogram is derived for the RPBART model to better understand how the specification of the prior hyperparameters impact the underlying model. For simplicity, consider the two dimensional input space $[-1, 1] \times [-1, 1]$. Figure 14 displays a more detailed example of the impact of the tree prior on the semivariogram. Each row of Figure 14 corresponds to a different level of smoothing, where the top row illustrates the lowest amount of smoothing and the bottom row illustrates the highest degree of smoothing. Each column corresponds to a different tree depth, where the leftmost column indicates deeper trees and the rightmost column indicates shallow trees.

Similar conclusions can be made to those in Section 3.4. Less smoothing and deeper trees result in a more localized fit (row 1 column 1 of Figure 14). In other words, less information is shared across the terminal nodes.

- For each row, the sill of the semivariogram shifts downward as the tree depth decreases. This is because shallower trees result in less terminal nodes. Less terminal nodes increases the probability that a pair of input points separated by a distance $||\boldsymbol{h}||$ are assigned to the same partition of data (and thus naturally more correlated).

- The value of k is inversely related to the prior variance for the terminal parameters, τ^2. In the semivariogram, τ^2 simply scales the covariance function. Thus, as k increases, the prior variance decreases and the semivariogram contracts. From a functional perspective, this means less variability in the data is attributed to the underlying mean (modeled as a sum-of-trees) and more variability will be attributed to the random error component.

- The smoothing parameters generally determine the size of the offset occurring when $||\boldsymbol{h}||$ is near 0. Low levels of smoothing result in a more localized fit. Thus, the correlation between nearby points assigned to adjacent terminal nodes is only non-zero for pairs near the cutpoints. The semivariogram is relatively linear in $||\boldsymbol{h}||$ when $||\boldsymbol{h}||$ is small (top row). This behavior of the semivariogram begins to resemble that of the exponential kernel, which is closely connected to tree kernels (Linero, 2017). As more smoothing is introduced, the correlation across points assigned to adjacent terminal nodes increases. In other words, pairs of points can have non-zero correlations even if they are further away from the cutpoints and assigned to different terminal nodes. This causes the offset to shift upwards. In this context, we refer to the "offset" as the value of the semivariogram as $||\boldsymbol{h}||$ approaches 0. This offset is always positive because of the discontinuity in the RPBART covariance function. Additionally, the higher smoothing case tends to result in a slight curvature in the tail, which is commonly observed with semivariogram that are defined by smooth kernels.

- An important theoretical note is that the RPBART model is only continuous in expectation. The discontinuity in the RPBART covariance results from the independence assumption between two random paths, $z_{bj}(\boldsymbol{x})$ and $z_{bj}(\tilde{\boldsymbol{x}})$. The discontinuity is emphasized when the prior emphasizes a more smooth expectation (rows 2 and 3).

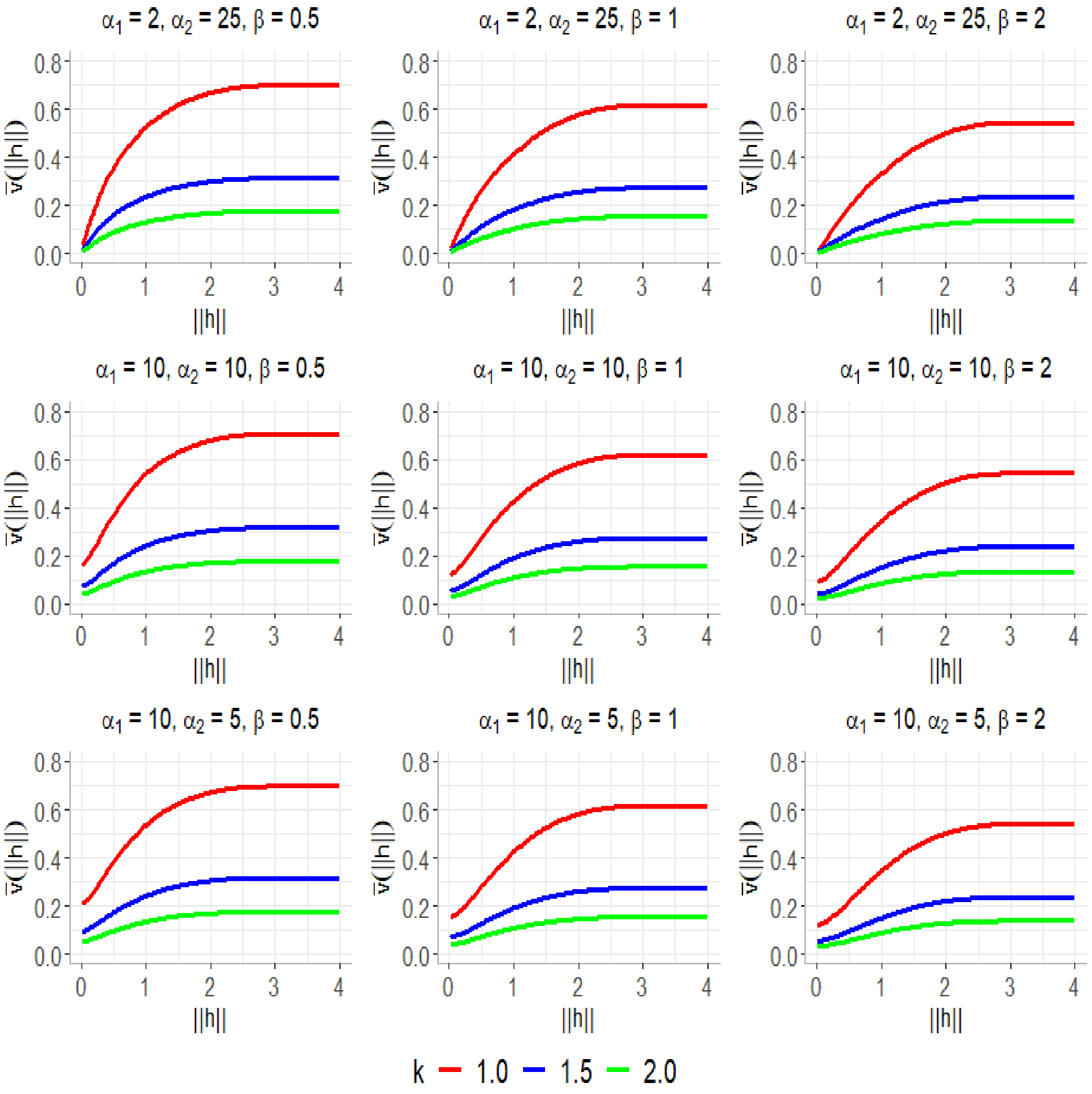


Figure 14: The semivariogram with respect to different hyperparameter settings for the tree, bandwidth, and terminal node parameter priors. Each row corresponds to different settings α_1 and α_2, which control the bandwidth. Each column corresponds to different values of β, which controls the tree size, and k. The three lines in each plot represent the semivariogram under different values of k. For simplicity, we set $y_{max} = 1$ and $y_{min} = -1$ in the computation of τ^2.

7.3.1 Application

We consider emulating the MIROC-ESL2 simulator to demonstrate the semivariogram in practice. This simulator is defined over the domain $[-90°, 90°]$ latitude and $[-180°, 180°]$ longitude with points spaced evenly along a 64×128 grid. For this emulation task, we consider the average monthly temperature output (degrees Celsius) for August 2014. The empirical semivariogram (gray points) along with the model-dependent theoretical semivariograms are shown in Figure 15. The top panel depicts the semivariogram for four possible selections of the RPBART prior. All priors emphasize lower levels of smoothing, which preserves a more localized fit with smoothing very close to a cutpoint. Low smoothing (and thus a more localized fit) is to be expected because this is an emulation task where almost all of the variability in the data should be attributed to the sum-of-trees rather than random error. In such a case, the offset value must be close to 0 for small $||\boldsymbol{h}||$.

The blue, green, and dark gray curves display the effects of the prior hyperparameters relative to the red curve. The red and dark gray curves compare the impact of α_2 on the prior. The gray curve involves relatively more smoothing (due to the smaller α_2 value), which results in a small increase in the offset and subtle curvature for $||\boldsymbol{h}||$ near 0. As $||\boldsymbol{h}||$ increases, the two semivariograms are nearly identical. Comparing the red and blue curves allows us to understand the impact of k, as larger k decreases the prior variance and thus contracts the semivariogram. Finally, comparing the red and green curves allows us to see the effect of the tree prior, where deeper trees (green) result in less correlation across neighboring points and thus a higher sill. These conclusions agree with the simulated semivariogram in Figure 14.

The bottom panel of Figure 15 displays three example fits for a semivariogram using an exponential kernel defined as $R(\boldsymbol{h}; s^2, \ell) = s^2 \exp(-||\boldsymbol{h}||/\ell)$. These different parameter choices were made simply to illustrate the similarities between the final semivariograms under the RPBART model and the exponential kernel.

Overall, we see similar functional forms for the semivariograms under the different models. Both the RPBART and the exponential kernels fit the empirical semivariogram fairly well for $||\boldsymbol{h}||$ between 0 and 150. The theoretical semivariograms deviate from the empirical semivariogram once reaching the sill. This is not uncommon, as in practice there are less pairs of points separated by large values of $||\boldsymbol{h}||$. Thus, the empirical semivariogram has greater uncertainty for large $||\boldsymbol{h}||$. This example not only illustrates how the RPBART semivariogram can compare to more common semivariograms, but also demonstrates practical challenges with the semivariogram. In practice, one can rarely obtain a perfect fit to the empirical semivariogram. Usually, the practitioner must select elements of the empirical data to fit. In this case, we focus on the fit to the left region of the empirical semivariogram, as we are particularly more concerned with the correlations between points that are nearby, rather than points that are much farther away (i.e. the correlation represented by the sill).

7.4 Extending the Semivariogram for Model Mixing

This section details how the RPBART semivariogram can be extended to model mixing.

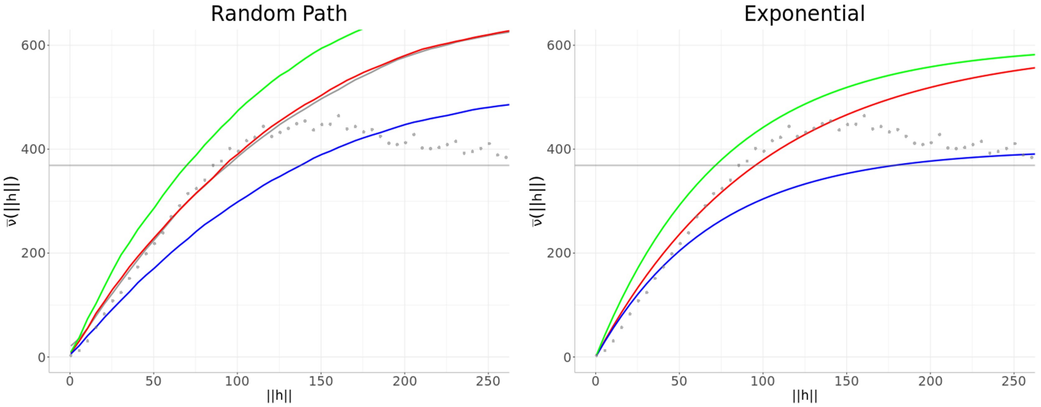


Figure 15: (Left) Four example semivariograms from the random path model using $k = 1.5, \alpha_1 = 2, \alpha_2 = 20, \beta = 1.75$ (dark gray) $k = 1.5, \alpha_1 = 2, \alpha_2 = 65, \beta = 1.75$ (red), $k = 1.7, \alpha_1 = 2, \alpha_2 = 65, \beta = 1.75$ (blue), and $k = 1.5, \alpha_1 = 2, \alpha_2 = 65, \beta = 1.0$ (green) (Right) Examples of the exponential semivariogram with covariance $R(\boldsymbol{x} + \boldsymbol{h}, \boldsymbol{x}; s^2, \ell) = s^2 \exp(-||\boldsymbol{h}||/\ell)$ with $(s^2, \ell) = (600, 100)$ (red), $(s^2, \ell) = (400, 70)$ (blue), and $(s^2, \ell) = (600, 75)$ (green). The empirical sill (horizontal gray) and empirical semivariogram (points) are displayed in both plots.

7.4.1 The Semivariogram formula for Model Mixing

To calibrate the remaining hyperparameters of our RPBART-based mixing model, we employ the prior semivariogram introduced in Section 3.4. The semivariogram of Theorem 3.1 can be extended to the model mixing framework by considering the assumptions associated with each emulator. One cannot directly apply the results conditional on the point estimates of each emulator, $\hat{f}_1(\boldsymbol{x}), \ldots, \hat{f}_K(\boldsymbol{x})$, because the semivariogram is designed to assess the spatial variability of the model free of any mean trend (Gringarten and Deutsch, 2001).

Rather than condition on the model predictions, we treat each of the individual emulators $f_1(\boldsymbol{x}), \ldots, f_K(\boldsymbol{x})$ as unknown quantities. Thus, the semivariogram for $Y(\boldsymbol{x})$ will assess the modeling choices for the sum-of-trees mixing model along with modeling choices for each $f_l(\boldsymbol{x})$. For example, we might assume GP emulators,

$$f_l(\boldsymbol{x}) \mid \boldsymbol{\psi}_l \sim \mathrm{GP}\big(\bar{f}_l,\ R_l(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\psi}_l)\big),$$

where $\boldsymbol{\psi}_l$ is a vector of parameters such as the scale or length scale and $\bar{f}_l$ is a constant mean.

Theorem 7.2 *Assume the random quantities $\{T_j, M_j, Z_j, \gamma_j\}_{j=1}^m$ are distributed according to Section 3.1. Assume each simulator is modeled as a stochastic emulator with mean $\bar{f}_l$*

and covariance kernel $R_l(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\psi}_l)$, $l = 1, \ldots, K$. Then, the function $\nu(\boldsymbol{x}, \boldsymbol{h})$ is

$$\nu(\boldsymbol{x}, \boldsymbol{h}) = \sigma^2 + \Big(\frac{1}{4k^2} + \frac{1}{K^2}\Big) \sum_{l=1}^{K} \nu_l^{(f)}(\boldsymbol{x}, h; \boldsymbol{\psi}_l)$$
$$+ \Big(\frac{1}{4k^2}\Big)\Big(1 - \bar{\Phi}(\boldsymbol{x}, \boldsymbol{h})\Big) \times \sum_{l=1}^{K} \Big(R_l(\boldsymbol{x} + \boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l) + \bar{f}_l^2\Big).$$

The semivariogram, $\overline{\nu}(\|\boldsymbol{h}\|)$, can be obtained by averaging over $\boldsymbol{x}$, as in (10). Rather than using the empirical semivariogram for $Y(\boldsymbol{x})$ to select the hyperparameters for the emulators and the weight functions, we recommend a modularization approach. Specifically, the hyperparameters associated with each emulator can be selected solely based on evaluations of the corresponding simulator output. The empirical semivariogram of $Y(\boldsymbol{x})$ can then be used to select the hyperparameters for the BART weights, plugging-in the choices of the $\boldsymbol{\psi}_l$ and $\bar{f}_l$ for $l = 1, \ldots, K$. Though this strategy is just used for hyperparameter selection, it aligns well with the two-step estimation procedure common in stacking (Breiman, 1996; Le and Clarke, 2017), which separates the information used to train the emulators from the observational data used to train the weights.

7.4.2 Derivation of the Semivariogram Formula for Model Mixing

Similar to Section 7.2, we first consider the function, $\nu(\boldsymbol{x}, \boldsymbol{h}; \Theta)$. In model mixing, this is derived by marginalizing over M_j, Z_j, and $f_l(\boldsymbol{x})$ where $j = 1, \ldots, m$ and $l = 1, \ldots, K$. For simplicity, assume each $f_l(\boldsymbol{x})$ is an independently distributed stochastic process with constant mean $\bar{f}_l$ and covariance function $R_l(\boldsymbol{x} + \boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l)$, with hyperparameter vector $\boldsymbol{\psi}_l$.

The function $\nu(\boldsymbol{x}, \boldsymbol{h}; \Theta)$ is defined in Theorem 7.3.

Theorem 7.3 *Assume the set of emulators, $f_1(\boldsymbol{x}), \ldots, f_K(\boldsymbol{x})$, are mutually independent random processes with constant means $\bar{f}_l$ and covariance functions $R_l(\boldsymbol{x} + \boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l)$, $l = 1, ..., K$. Further assume the emulators are independent of the K weight functions. Then, the function $\nu(\boldsymbol{x}, \boldsymbol{h}; \Theta)$ for the mean-mixing model is given by*

$$\nu(\boldsymbol{x}, \boldsymbol{h}; \Theta) = \sigma^2 + \Big(m\tau^2 + \frac{1}{K^2}\Big) \sum_{l=1}^{K} \nu_l^{(f)}(\boldsymbol{x}, h; \boldsymbol{\psi}_l)$$
$$+ \nu^{(w)}(\boldsymbol{x}, h; \Theta) \sum_{l=1}^{K} \Big(R_l(\boldsymbol{x} + \boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l) + \bar{f}_l^2\Big)$$

where $\nu_l^{(f)}(\boldsymbol{x}, h; \boldsymbol{\psi}_l) = R_l(\boldsymbol{x}, \boldsymbol{x}; \boldsymbol{\psi}_l) - R_l(\boldsymbol{x} + \boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l)$ is semivariogram component from the lth emulator, $R(\boldsymbol{x} + \boldsymbol{h}, \boldsymbol{x} + \boldsymbol{h}; \boldsymbol{\psi}_l) = R(\boldsymbol{x}, \boldsymbol{x}; \boldsymbol{\psi}_l)$, and

$$\nu^{(w)}(\boldsymbol{x}, h; \Theta) = m\tau^2\Big(1 - \frac{1}{m} \sum_{j=1}^{m} \sum_{b=1}^{B_j} \phi_{bj}(\boldsymbol{x} + \boldsymbol{h}; T_j, \gamma_j)\phi_{bj}(\boldsymbol{x}; T_j, \gamma_j)\Big)$$

is the semivariogram component from each of the K weight functions.

Proof

By definition, the function $\nu(\boldsymbol{x}, \boldsymbol{h}; \Theta)$ is defined by

$$\begin{aligned}\nu(\boldsymbol{x}, \boldsymbol{h}; \Theta) &= \frac{1}{2}\mathrm{Var}\Big(Y(\boldsymbol{x}+\boldsymbol{h}) - Y(\boldsymbol{x}) \mid \Theta\Big)\\ &= \frac{1}{2}\Big(\mathrm{Var}\Big(Y(\boldsymbol{x}+\boldsymbol{h}) \mid \Theta\Big) + \Big(Y(\boldsymbol{x}) \mid \Theta\Big)\Big)\\ &\quad - \mathrm{Cov}\big(Y(\boldsymbol{x}+\boldsymbol{h}) - Y(\boldsymbol{x})\end{aligned}$$

The conditional variance simplifies as,

$$\begin{aligned}\mathrm{Var}\Big(Y(\boldsymbol{x}) \mid \Theta\Big) &= \mathrm{Var}\Big(\sum_{l=1}^{K} w_l(\boldsymbol{x}) f_l(\boldsymbol{x}) + \epsilon(\boldsymbol{x}) \mid \Theta\Big)\\ &= \sigma^2 + \sum_{l=1}^{K} \mathrm{Var}\Big(w_l(\boldsymbol{x}) f_l(\boldsymbol{x}) \mid \Theta\Big),\end{aligned}$$

where the K weights and K functions are all mutually independent, conditional on Θ. Each individual variance component then simplifies as

$$\begin{aligned}\mathrm{Var}\Big(w_l(\boldsymbol{x}) f_l(\boldsymbol{x}) \mid \Theta\Big) &= E\Big[w_l(\boldsymbol{x})^2 f_l(\boldsymbol{x})^2 \mid \Theta\Big] - E\Big[w_l(\boldsymbol{x}) f_l(\boldsymbol{x}) \mid \Theta\Big]^2\\ &= E\big[w_l(\boldsymbol{x})^2 \mid \Theta\big] E\big[f_l(\boldsymbol{x})^2\big] - E\big[w_l(\boldsymbol{x}) \mid \Theta\big]^2 E\big[f_l(\boldsymbol{x})\big]^2\\ &= \Big(m\tau^2 + 1/K^2\Big)\Big(R_l(\boldsymbol{x}, \boldsymbol{x}; \boldsymbol{\psi}_l) + \bar{f}_l^2\Big) - \Big(\bar{f}_l/K\Big)^2.\end{aligned}$$

Similarly, the conditional variance of $w_l(\boldsymbol{x}+\boldsymbol{h}) f_l(\boldsymbol{x}+\boldsymbol{h})$ is expressed by

$$\begin{aligned}\mathrm{Var}\Big(w_l(\boldsymbol{x}+\boldsymbol{h}) f_l(\boldsymbol{x}+\boldsymbol{h}) \mid \Theta\Big) &= \Big(m\tau^2 + 1/K^2\Big)\\ &\quad \times \Big(R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}+\boldsymbol{h}; \boldsymbol{\psi}_l) + \bar{f}_l^2\Big)\\ &\quad - \Big(\bar{f}_l/K\Big)^2,\end{aligned}$$

for $l = 1, \ldots, K$.

The conditional covariance can then be computed as

$$\begin{aligned}\mathrm{Cov}\Big(Y(\boldsymbol{x}+\boldsymbol{h}), Y(\boldsymbol{x}) \mid \Theta\Big) &= \mathrm{Cov}\Big(\sum_{l=1}^{K} w_l(\boldsymbol{x}+\boldsymbol{h}) f_l(\boldsymbol{x}+\boldsymbol{h}),\ \sum_{t=1}^{K} w_t(\boldsymbol{x}) f_t(\boldsymbol{x}) \mid \Theta\Big)\\ &= \sum_{l=1}^{K}\sum_{t=1}^{K} \mathrm{Cov}\Big(w_l(\boldsymbol{x}+\boldsymbol{h}) f_l(\boldsymbol{x}+\boldsymbol{h}),\ w_t(\boldsymbol{x}) f_t(\boldsymbol{x}) \mid \Theta\Big).\end{aligned}$$

Due to conditional independence between the K weight functions and the K emulators, the covariance simplifies as

$$\begin{aligned}
\mathrm{Cov}\Big(Y(\boldsymbol{x}+\boldsymbol{h}), Y(\boldsymbol{x}) \mid \Theta\Big) &= \sum_{l=1}^{K} \mathrm{Cov}\Big(w_l(\boldsymbol{x}+\boldsymbol{h}) f_l(\boldsymbol{x}+\boldsymbol{h}),\ w_l(\boldsymbol{x}) f_l(\boldsymbol{x}) \mid \Theta\Big) \\
&= \sum_{l=1}^{K} E\Big[w_l(\boldsymbol{x}+\boldsymbol{h}) f_l(\boldsymbol{x}+\boldsymbol{h}) w_l(\boldsymbol{x}) f_l(\boldsymbol{x}) \mid \Theta\Big] \\
&\qquad - E\Big[w_l(\boldsymbol{x}+\boldsymbol{h}) f_l(\boldsymbol{x}+\boldsymbol{h}) \mid \Theta\Big] E\Big[w_l(\boldsymbol{x}) f_l(\boldsymbol{x}) \mid \Theta\Big] \\
&= \sum_{l=1}^{K} E\Big[w_l(\boldsymbol{x}+\boldsymbol{h}) w_l(\boldsymbol{x}) \mid \Theta\Big] E\Big[f_l(\boldsymbol{x}+\boldsymbol{h}) f_l(\boldsymbol{x})\Big] - \Big(\bar{f}_l/K\Big)^2 \\
&= \sum_{l=1}^{K} \big(m\tau^2 \bar{\phi}(\boldsymbol{x}, h; \Theta) + 1/K^2\big)\Big(R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l) + \bar{f}_l^2\Big) \\
&\qquad - \sum_{l=1}^{K} \Big(\bar{f}_l/K\Big)^2.
\end{aligned}$$

where $\bar{\phi}(\boldsymbol{x}, h; \Theta) = \frac{1}{m}\sum_{j=1}^{m}\sum_{b=1}^{B_j} \phi_{bj}(\boldsymbol{x}+\boldsymbol{h}; T_j, \gamma_j)\phi_{bj}(\boldsymbol{x}; T_j, \gamma_j)$. Using the conditional variances and covariance, $\nu(\boldsymbol{x}, \boldsymbol{h}; \Theta)$ can be expressed as

$$\begin{aligned}
\nu(\boldsymbol{x}, \boldsymbol{h}; \Theta) &= \frac{1}{2}\Bigg(\sigma^2 + \sum_{l=1}^{K}\Big(R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}+\boldsymbol{h}; \boldsymbol{\psi}_l) + \bar{f}_l^2\Big)\Big(m\tau^2 + 1/K^2\Big) - \Big(\bar{f}_l/K\Big)^2\Bigg) \\
&\quad + \frac{1}{2}\Bigg(\sigma^2 + \sum_{l=1}^{K}\Big(R_l(\boldsymbol{x}, \boldsymbol{x}; \boldsymbol{\psi}_l) + \bar{f}_l^2\Big)\Big(m\tau^2 + 1/K^2\Big) - \Big(\bar{f}_l/K\Big)^2\Bigg) \\
&\quad - \Bigg(\sum_{l=1}^{K}\big(m\tau^2\bar{\phi}(\boldsymbol{x}, h; \Theta) + 1/K^2\big)\Big(R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l) + \bar{f}_l^2\Big) - \Big(\bar{f}_l/K\Big)^2\Bigg)
\end{aligned}$$

This expression can then be simplified as follows:

$$
\begin{aligned}
\nu(\boldsymbol{x}, \boldsymbol{h}; \Theta) = \sigma^2 &+ \frac{1}{2}\Big(\sum_{l=1}^{K} \Big(R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}+\boldsymbol{h}; \boldsymbol{\psi}_l) \Big) \Big(m\tau^2 + 1/K^2 \Big) + m\tau^2 \bar{f}_l^2 \Big) \\
&+ \frac{1}{2}\Big(\sum_{l=1}^{K} \Big(R_l(\boldsymbol{x}, \boldsymbol{x}; \boldsymbol{\psi}_l) \Big) \Big(m\tau^2 + 1/K^2 \Big) + m\tau^2 \bar{f}_l^2 \Big) \\
&- \Bigg(\sum_{l=1}^{K} \big(m\tau^2 \bar{\phi}(\boldsymbol{x}, h; \Theta) + 1/K^2 \big) R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l) + m\tau^2 \bar{f}_l^2 \bar{\phi}(\boldsymbol{x}, h; \Theta) \Bigg) \\
= \sigma^2 &+ \frac{1}{2}\Big(m\tau^2 + 1/K^2 \Big)\Big(\sum_{l=1}^{K} \Big(R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}+\boldsymbol{h}; \boldsymbol{\psi}_l) + R_l(\boldsymbol{x}, \boldsymbol{x}; \boldsymbol{\psi}_l) \Big) \\
&+ m\tau^2 \big(1 - \bar{\phi}(\boldsymbol{x}, h; \Theta) \big) \sum_{l=1}^{K} \bar{f}_l^2 \\
&- \big(m\tau^2 \bar{\phi}(\boldsymbol{x}, h; \Theta) + 1/K^2 \big) \sum_{l=1}^{K} R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l) \\
&\pm \Big(m\tau^2 + 1/K^2 \Big) \sum_{l=1}^{K} R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l),
\end{aligned}
$$

where we can add and subtract $\Big(m\tau^2 + 1/K^2\Big) \sum_{l=1}^{K} R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l)$ to further simplify the expression. Now define the following functions for each emulator and each weight as

$$
\nu_l^{(f)}(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\psi}_l) = \frac{1}{2}\Big(R_l(\boldsymbol{x}, \boldsymbol{x}; \boldsymbol{\psi}_l) + R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}+\boldsymbol{h}; \boldsymbol{\psi}_l) - 2R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l) \Big) \tag{21}
$$

$$
\nu^{(w)}(\boldsymbol{x}, \boldsymbol{h}; \Theta) = m\tau^2 \Big(1 - \bar{\phi}(\boldsymbol{x}, h; \Theta) \Big) \tag{22}
$$

In typical cases, $R(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}+\boldsymbol{h}; \boldsymbol{\psi}_l) = R(\boldsymbol{x}, \boldsymbol{x}; \boldsymbol{\psi}_l)$ and thus $\nu_l^{(f)}(\boldsymbol{x}, h; \boldsymbol{\psi}_l)$ simplifies further. The terms in the expression can then be rearranged as follows

$$
\begin{aligned}
\nu(\boldsymbol{x}, \boldsymbol{h}; \Theta) = \sigma^2 &+ \Big(m\tau^2 + 1/K^2 \Big) \sum_{l=1}^{K} \nu_l^{(f)}(\boldsymbol{x}, \boldsymbol{h}; \boldsymbol{\psi}_l) \\
&+ \nu^{(w)}(\boldsymbol{x}, \boldsymbol{h}; \Theta) \Big(\sum_{l=1}^{K} R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l) + \bar{f}_l^2 \Big).
\end{aligned}
$$

■

Theorem 7.3 shows the function $\nu(\boldsymbol{x}, \boldsymbol{h}; \Theta)$ combines information from the model set and the weight functions. More specifically, we see the expression combines the semivariogram functions from the $K+1$ different components. Using Theorem 7.3, we can compute the $\nu(\boldsymbol{x}, \boldsymbol{h})$ in Definition 3.1 by marginalizing over the set of m trees and bandwidth parameters. The resulting function is given by

$$\begin{aligned}\nu(\boldsymbol{x}, \boldsymbol{h}) = \sigma^2 &+ \Big(\frac{1}{4k^2} + \frac{1}{K^2}\Big) \sum_{l=1}^{K} \nu_l^{(f)}(\boldsymbol{x}, h; \boldsymbol{\psi}_l) \\ &+ \Big(\frac{1}{2k}\Big)^2 \Big(1 - E\Big[\sum_{b=1}^{B_1} \phi_{b1}(\boldsymbol{x}+\boldsymbol{h}; T_1, \gamma_1)\phi_{b1}(\boldsymbol{x}; T_1, \gamma_1)\Big]\Big) \\ &\times \sum_{l=1}^{K} \Big(R_l(\boldsymbol{x}+\boldsymbol{h}, \boldsymbol{x}; \boldsymbol{\psi}_l) + \bar{f}_l^2\Big).\end{aligned}$$

■

We note, the proof of Theorem 7.2 follows directly from Theorem 7.3 with the additional step of taking the expectation with respect to Θ.

7.5 Algorithm

The sampling algorithm for the random path model is given by Algorithm 1.

Algorithm 1 The sampling algorithm for the random path model.

```
Data: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$
Result: Approximate posterior samples drawn from $\pi\Big(\big\{(T_j, M_j, Z_j, \gamma_j)\big\}_{j=1}^m, \sigma^2 \mid \boldsymbol{y}\Big)$
  for $N_{mcmc}$ iterations do
      for $j = 1, \ldots, m$ do
          (1) With probability $p_{\text{birth}}$, draw $T_j$ and partially update $Z_j$ using RJMCMC.
              Otherwise update via a death step.
          (2) Update each splitting rule $(v_{dj}, c_{dj})$ individually for $d = 1, \ldots, B_j - 1$,
              where $T_j$ has $B_j - 1$ internal nodes.
          (3) Draw $Z_j \mid T_j, \gamma_j, \boldsymbol{r}_j, \sigma^2$
          (4) Draw $M_j \mid T_j, Z_j, \gamma_j, \boldsymbol{r}_j, \sigma^2$
          (5) Draw $\gamma_j \mid T_j, M_j, Z_j, \boldsymbol{r}_j, \sigma^2$
      end for
      (6) Draw $\sigma^2 \mid \boldsymbol{y}, \big\{(T_j, M_j), Z_j, \gamma_j\big\}_{j=1}^m$
  end for
```

Algorithm 1 involves conditional sampling steps for each of the components of the RPBART model. Two main differences exist between the RPBART algorithm and previous BART sampling algorithms (Pratola, 2016; Chipman et al., 2010). The first difference is in updating the tree structure, as the dimension of Z_j will also change as the dimension of T_j changes. Each tree, T_j, can be updated via a birth step, which adds one new terminal node, or a death step, which removes a pair of terminal nodes. Both of these moves will have analogous effects on the dimension the n vectors within Z_j; a birth step will increase the dimension by one, while a death step will decrease the dimension by one. A simple reversible jump MCMC (RJMCMC) is used to navigate the dimension change within Z_j when jointly updating

$$(T_j, Z_j) \to (T'_j, Z'_j)$$

where T'_j is the proposed tree and Z'_j is the set of proposed random path vectors.

7.5.1 Birth Step:

Without loss of generality, consider a birth proposal that splits the first terminal node $\eta^{(t)}_{1j}$ using the rule $x_v < c$. The proposed move adds one more dimension to each random path vector. Thus, we define the proposed set Z'_j by

$$\begin{aligned} Z'_j &= \Big\{ \boldsymbol{z}'_j(\boldsymbol{x}_i) \Big\}_{i=1}^{n} \\ \boldsymbol{z}'(\boldsymbol{x}_i) &= \Big(z^{(l)}_{1j}(\boldsymbol{x}_i),\ z^{(r)}_{1j}(\boldsymbol{x}_i),\ z'_{2j}(\boldsymbol{x}_i), \ldots, z'_{B_j j}(\boldsymbol{x}_i) \Big), \end{aligned}$$

where $z^{(l)}_{1j}(\boldsymbol{x}_i)$ and $z^{(r)}_{1j}(\boldsymbol{x}_i)$ are the new random path indicators for the proposed left and right child nodes. The other components $z'_{bj}(\boldsymbol{x}_i)$, $b = 2, \ldots, B_j$, are the random path indicators corresponding to the terminal nodes which are unaffected by the birth proposal.

Since the tree proposal only affects the first terminal node, we assume the proposal to the random path vectors only affects the first component of the vector. As a result, we set $z'_{bj}(\boldsymbol{x}_i) = z_{bj}(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$ and $b = 2, \ldots, B_j$ and only consider an update for the first component $z_{1j}(\boldsymbol{x}_i)$. The proposed move can be simplified as

$$\Big(T_j,\ \{z_{1j}(\boldsymbol{x}_i)\}_{i=1}^{n}\Big) \ \to\ \Big(T'_j,\ \{z^{(l)}_{1j}(\boldsymbol{x}_i),\ z^{(r)}_{1j}(\boldsymbol{x}_i)\}_{i=1}^{n}\Big), \tag{23}$$

where $z_{1j}(\boldsymbol{x}_i)$ is the random path indicator for $\eta^{(t)}_{1j}$.

To facilitate the RJMCMC required by (23), we introduce a set of n independent augmented random variables, $u_{1j}, \ldots, u_{nj}$, each of which can be interpreted as the event that the ith observation is randomly mapped to the new right child node. A simple strategy is to set $u_{ij} = 1$ if $x_{iv} \geq c_v$ and $u_{ij} = 0$ otherwise. This strategy imposes a deterministic split to assess the impact of adding a new terminal node to the tree without any additional stochasticity of the random path assignments. A more generalized strategy is further discussed in (Yannotty, 2024).

The augmented random variables are then used to match the dimensions in the proposal as shown in (26),

$$z^{(r)}_{1j}(\boldsymbol{x}_i) = u_{ij}\ z_{1j}(\boldsymbol{x}_i) \tag{24}$$

$$z^{(l)}_{1j}(\boldsymbol{x}_i) = (1 - u_{ij})\ z_{1j}(\boldsymbol{x}_i) \tag{25}$$

$$z'_{bj}(\boldsymbol{x}_i) = z_{bj}(\boldsymbol{x}_i), \quad b = 2, \ldots, B_j. \tag{26}$$

7.5.2 Death Step:

The death proposal can be viewed as the reversal of the birth proposal. Without loss of generality, assume T'_j has $B_j + 1$ terminal nodes denoted by $\eta^{(l)}_{1j}$, $\eta^{(r)}_{1j}$, and $\eta^{(t)}_{bj}$ for $b =$

$2, \dots, B_j$. Now suppose the death move reconstructs T_j by replacing $\eta_{1j}^{(l)}$ and $\eta_{1j}^{(r)}$ with $\eta_{1j}^{(t)}$. This proposal is summarized by

$$\Big(T_j',\ \big\{z_{1j}^{(l)}(\boldsymbol{x}_i),\ z_{1j}^{(r)}(\boldsymbol{x}_i)\big\}_{i=1}^{n}\Big) \ \rightarrow\ \Big(T_j,\ \big\{z_{1j}(\boldsymbol{x}_i)\big\}_{i=1}^{n}\Big). \tag{27}$$

Once more, we wish to fix the remainder of the tree in T_j' and focus solely on the death proposal at the specified terminal nodes. In such a case, no augmentation is needed and the deterministic map is given by

$$\begin{aligned} z_{1j}(\boldsymbol{x}_i) &= z_{1j}^{(r)}(\boldsymbol{x}_i) + z_{1j}^{(l)}(\boldsymbol{x}_i) \\ z_{bj}(\boldsymbol{x}_i) &= z_{bj}'(\boldsymbol{x}_i), \quad b = 2, \dots, B_j \end{aligned}$$

for $i = 1, \dots, n$. This implies that only the observations that are mapped to either the left or right children under consideration will affect the evaluation of the death proposal.

7.5.3 Updating Z_j

Given the update to the tree in Step 1, we further update the random path assignments using a Metropolis-Hastings algorithm. In such a case, we simply use the prior distribution as a proposal distribution. In some cases, a modularized approach can be used to obtain more efficient updating of Z_j (Yannotty, 2024; Liu et al., 2009). The modularized approach ignores the information in the backfitting residuals (which is used to update all other elements of the model) and simply updates Z_j by only considering the current information in Z_j, the selected proposal distribution, the updated values of the bandwidth parameters, and the updated tree structure.

7.6 Supporting Examples

This section provides additional comparisons between RPBART and other competing methods in both regression and model mixing tasks.

7.6.1 RPBART Regression Example

Consider the true data generating process

$$\begin{aligned} Y(\boldsymbol{x}) &\sim N\big(f_\dagger(\boldsymbol{x}), \sigma^2\big) \\ f_\dagger(\boldsymbol{x}) &= \sin(x_1) + \cos(x_2) \end{aligned}$$

where $\boldsymbol{x} \in [-\pi, \pi] \times [-\pi, \pi]$. Assume $n = 100$ different observations are generated from the true underlying process with $\sigma = 0.1$. The inputs $\boldsymbol{x}_1, \dots, \boldsymbol{x}_{100}$ are also randomly generated about the two-dimensional input space. We train the random path model using three different settings of the bandwidth prior to demonstrate the smoothing effect of the model. Each of the three models are trained using an ensemble of 20 trees and a value of $k = 1$.

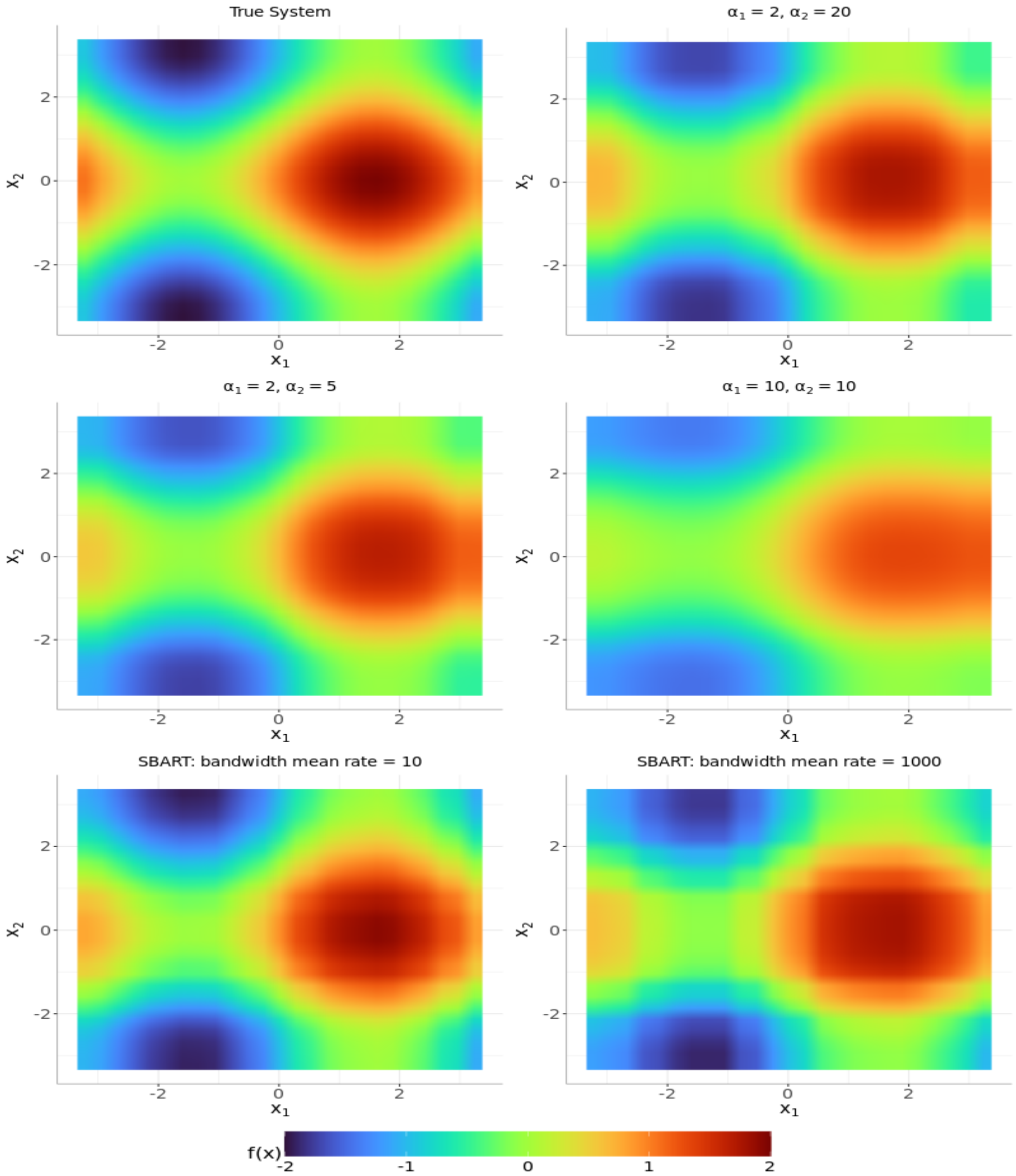


Figure 16: The true system (top left) compared to the the posterior mean predictions from RPBART with different settings of the bandwidth prior as indicated by the values of α_1 and α_2 (top right and middle row). Two mean predictions from SBART are included with a bandwidth mean rates of 10 and 1000, respectively (bottom row).

Figure 16 displays the mean predictions from the RPBART model along with their associated hyperparameters of α_1 and α_2. Based on Figure 16, we see the model with less smoothing (i.e. $\alpha_1 = 2$ and $\alpha_2 = 20$) provides the most accurate predictions across the entire domain. Essentially, this model preserves the localization induced by the tree models but is able to smooth over the hard boundaries. With $\alpha_1 = 2$ and $\alpha_2 = 5$, we see the a similar smooth mean surface, however the prediction is slightly less accurate along the boundaries (particularly when x_1 is near $-\pi$). We note BART typically struggles along the boundaries when no or minimal data is present, so this occurrence is to be expected. Similar observations can be made as more smoothness is introduced into the model as shown in the middle right panel of Figure 16, where $\alpha_1 = 10$ and $\alpha_2 = 10$.

These results are to be expected as higher levels of smoothing will naturally result in a less localized prediction. Despite this, RPBART is able to consistently identify the main features of the underlying process, regardless the level of smoothing. In cases with more smoothing, the mean function will identify the general pattern of the data and attribute some of the granularity in the process to the observational error.

For comparison, two mean predictions from the SBART model are also displayed in the bottom row of Figure 16. In SBART, the bandwidth parameter is assigned an exponential prior with rate parameterization. Thus, higher values of the bandwidth lead to less smoothing. In terms of predictive performance, we observe SBART is slightly more accurate in predicting the peaks (dark red) and valleys (dark blue) of the function. Despite this, the mean prediction from SBART still exhibits a block-like structure, which is commonly observed in BART due to the discontinuity of the trees. This most notably occurs near the peak centered around $(2, 0)$. Finally, the two SBART predictions are fairly similar, which suggests the model is less sensitive to the prior specification compared to RPBART. Meanwhile, RPBART produces a smoother mean prediction across the domain. The degree of smoothing can be easily controlled through the prior distribution.

7.6.2 Comparing BART-BMM and RPBART-BMM

In this section, we revisit the 2-dimensional Taylor series mixing example by comparing the RPBART-BMM and BART-BMM results across 50 different datasets simulated from the same underlying system. Figure 17 compares the distribution of the prediction RMSE of BART-BMM versus the prediction RMSE of RPBART-BMM. Clearly, RPBART-BMM outperforms BART-BMM across the simulation study. Ultimately, RPBART-BMM is less prone to overfitting in the region where both simulators are inaccurate and minimal data is available.

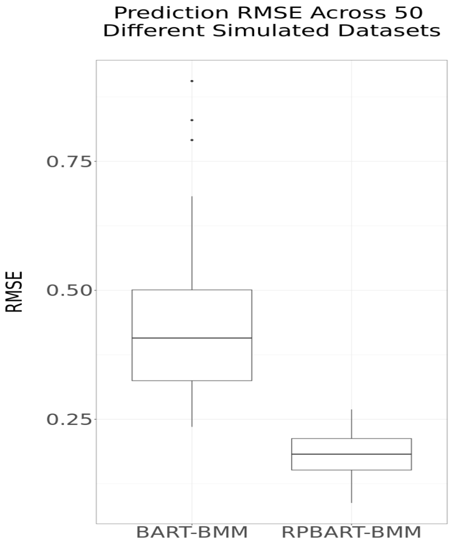


Figure 17: The distribution of the prediction RMSE for BART-BMM and RPBART-BMM across 50 different simulated datasets.

7.6.3 Climate Results

This section formally compares the results of the three mixing methods (RPBART-BMM, NNS, and FWLS) and a simple average of the 8 GCMs to those of the individual GCMs. Table 2 illustrates that each mixing approach substantially improves prediction accuracy compared to an individual GCM.

Model	April 2014	August 2014	December 2014
RPBART-BMM	**0.827**	**0.864**	**0.882**
NNS	0.993	1.032	1.088
FWLS	2.393	2.345	2.142
Average	2.826	3.089	2.960
Access	3.675	3.813	4.802
BCC	4.229	4.320	4.340
MIROC	5.529	5.590	6.979
CMCC	4.454	4.272	6.369
CESM2	3.230	3.681	3.419
CNRM	3.289	3.053	3.580
Can-ESM5	3.665	3.977	3.759
KIOST	3.763	4.645	3.639

Table 2: The prediction root mean square error for each mixing approach (RPBART-BMM, NNS, and FWLS), an average of the 8 GCMs, and the individual GCMs (remaining rows) stratified by month. Each RMSE is computed by evaluating the associated model-based prediction over a dense grid of 259,200 inputs and comparing to the observed ERA5 data. The smallest RMSEs for each month are denoted in bold.