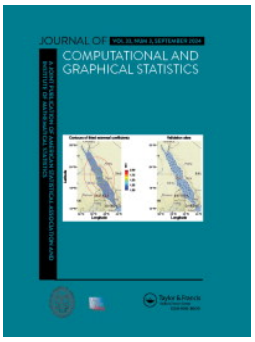

Taylor & Francis
Taylor & Francis Group

Journal of Computational and Graphical Statistics

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/ucgs20

AddiVortes: (Bayesian) Additive Voronoi Tessellations

Adam. J. Stone & John Paul Gosling

To cite this article: Adam. J. Stone & John Paul Gosling (16 Oct 2024): AddiVortes: (Bayesian) Additive Voronoi Tessellations, Journal of Computational and Graphical Statistics, DOI: 10.1080/10618600.2024.2414104

To link to this article: https://doi.org/10.1080/10618600.2024.2414104

AddiVortes: (Bayesian) Additive Voronoi Tessellations

Adam. J. Stone* and John Paul Gosling

Department of Mathematical Sciences, Durham University, UK

*adam.stone2@durham.ac.uk

Abstract

The Additive Voronoi Tessellations (AddiVortes) model is a multivariate regression model that uses Voronoi tessellations to partition the covariate space in an additive ensemble model. Unlike other partition methods, such as decision trees, this has the benefit of allowing the boundaries of the partitions to be non-orthogonal and non-parallel to the covariate axes. The AddiVortes model uses a similar sum-of-tessellations approach and a Bayesian backfitting MCMC algorithm to the BART model. We utilize regularization priors to limit the strength of individual tessellations and accepts new models based on a likelihood. The performance of the AddiVortes model is illustrated through testing on several data sets and comparing the performance to other models along with a simulation study to verify some of the properties of the model. In many cases, the AddiVortes model outperforms random forests, BART and other leading black-box regression models when compared using a range of metrics. Supplementary materials for this article are available online.

Keywords: Bayesian Methods, Black Box Algorithm, Multivariate Analysis, Nonparametric Regression, Voronoi Tessellations.

1 Introduction

Many methods aim to model a conditional expectation function *f* that relates the continuous variable *Y* to some or all of *p* (potential) covariates $\boldsymbol{x} = (x_1, \ldots, x_p)$, such that

$$Y = f(\boldsymbol{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2),$$

under the assumption that the noise is mean zero and homoscedastic. In the present paper, we model the systematic part of this relationship using a sum of step-wise functions built upon Voronoi tessellations. More specifically, we approximate $f(\boldsymbol{x}) = \mathbb{E}(Y \mid \boldsymbol{x})$, the mean of *Y* given $\boldsymbol{x}$, by the sum of many piecewise constant functions with boundaries defined by Voronoi tessellations.

The use of Voronoi tessellations can be traced back to Descartes in 1644, but the first noteworthy use was in Dirichlet (1850) where two-dimensional and three-dimensional Voronoi tessellations (Dirichlet diagrams) were used in his research on quadratic forms. These tessellations were later named after Georgy Voronoi, who extended them to the general *n*-dimensional case in Voronoi (1908). Voronoi tessellations have applications in many areas such as computer science (for example, Musgrave et al., 1989; Shewchuk, 2002; Galceran and Carreras, 2013) and biology (for example, Bock et al., 2009; Li et al., 2012). More recently, in Pope et al. (2021), Voronoi tessellations were employed in a reversible-jump MCMC (Monte Carlo Markov Chain) algorithm by modifying centers to partition the covariate space and perform separate Gaussian processes over different regions to model high dimensional non-smooth functions.

In this paper, we utilize Voronoi tessellations as part of an ensemble algorithm for naturally partitioning the covariate space. An ensemble method is a kind of a “meta-algorithm” that aggregates over several potentially weaker sub-models to produce one high-performing predictive model. These are powerful methods because ensemble techniques can improve prediction accuracy, handle complexity and offer interpretability. However, these methods have high computational expense and require careful tuning of the hyperparameters. Some popular ensemble methods include bagging (Breiman, 1996), boosting (Freund and Schapire, 1997) and random forests (Breiman, 2001). Boosting builds a sequence of trees where each tree corrects the errors made by the previous ones, while bagging and random forests use randomness to generate a large collection of individual trees and average their predictions to make them more robust.

An additive approach is an ensemble technique that makes predictions by summing the outputs of individual models. The BART model, introduced by Chipman et al. (2010), is an example of such a method, and is referred to as a sum-of-trees model in the paper. In the BART model, several decision trees’ outputs are summed together to provide a prediction and the advantage of this approach lies in its ability to capture both the main effects, nonlinearities and interaction effects of the covariates. In Bayesian additive models, to prevent any single model from exerting too much influence and to mitigate over-fitting, regularization priors are employed. Regularization priors limit the complexity of a single model. In the case of the BART model, these priors are used to constrain the depth of each decision tree, thereby reducing over-fitting.

In this paper, we introduce the Additive Voronoi Tessellation (AddiVortes) model. It is an ensemble method that models *m* (Voronoi) tessellations by using a reversible-jump MCMC backfitting algorithm to iteratively update each tessellation. An additive approach is used so each tessellation captures a fraction of the overall prediction since the outputs of each are added together to give the expected value of the dependent variable. Regularization priors are used to reduce the likelihood of individual tessellations having too many centers and dimensions and thus reducing over-fitting.

To facilitate the use of the AddiVortes methods described in this paper, we have provided a repository on GitHub with all the code to run the algorithm and produce all the figures in this paper. It is available at https://github.com/Adam-Stone2/AddiVortes.

The remainder of the paper is organized as follows. In Section 2, the fundamental concepts which AddiVortes is built upon are outlined. In Section 3, a Bayesian backfitting MCMC algorithm and methods for inference are described. In Sections 4 and 5, we illustrate the potential of AddiVortes through a diverse range of examples, encompassing both simulated scenarios and real-world data. Section 7 concludes the paper with a discussion and possible extensions to the AddiVortes algorithm.

2 Fundamentals of the AddiVortes model

In general, a Voronoi tessellation partitions a metric space $(\mathcal{S}, d)$. In this paper, tessellations partition subsets of the covariate space using the Euclidean distance. However, the algorithm can be easily modified to accommodate other metrics. Given a set of *b* distinct points $\boldsymbol{M} = \{c_1, c_2, \ldots, c_b\}$ in the metric space, a Voronoi cell V_i, associated with the center c_i, is defined as:

$$V_i = \{\boldsymbol{x} \in \mathcal{S} : d(\boldsymbol{x}, c_i) \leq d(\boldsymbol{x}, c_j) \text{ for all } j \neq i\}.$$

In other words, a point in the space is in a cell associated with a center if it is closest in distance (with respect to the given distance metric) to that center compared to the others.

For AddiVortes, each tessellation incorporates unique covariates as dimensions, i.e. if the j^{th} tessellation includes 3 covariates then it is 3 dimensional. Tessellations have parameters $\mu_{ij} \in \mathbb{R}$ for each of the $i = 1, \ldots, b_j$ cells in the j^{th} tessellation and this value is referred to as the output value for all the samples contained within that cell. For instance, in Figure 1, the tessellation includes the covariates x_1 and x_2 so is 2 dimensional and the points represent observations which correspond to the output value for the cell they are in. The set of all output values for the j^{th} tessellation is denoted by $\boldsymbol{M}_j = \{\mu_{1j}, \ldots \mu_{b_j j}\}$.

For ease of computation, like in Chipman et al. (2010), we scale and transform the dependent variable *Y* such that the minimum and maximum values of $\boldsymbol{y}_{train}$ are -0.5 and 0.5, respectively, i.e. $\min\{y_{\text{train}}\} = -0.5$ and $\max\{y_{\text{train}}\} = 0.5$. Similarly, to determine appropriate prior distributions for the coordinates of the centers, we linearly scale and shift the covariates so that their minimum and maximum values of the training data are at -0.5 and 0.5, respectively.

2.1 Categorical data

To allow nominal categorical data to be used in AddiVortes, we convert categorical variables into numerical variables. There are several established methods to deal with categorical data, and a review of different techniques can be found in Kosaraju et al. (2023) where they applied categorical encoding to a Heart Disease Prediction data set. We recommend two methods depending on the number of categorical variables and categories within each variable.

If there are a small number of categories and a small number of categorical variables, then we may use the one-hot method to encode the categorical variables. This involves treating each category as a new variable and assigning the sample a value of one if the sample falls into that category and zero otherwise. An obvious problem with the one-hot method is that if there are many categories then the number of variables in the model increases significantly. This will increase the uncertainty in choosing the correct variables in the tessellations that are the most influential on the outcome variable.

Alternatively, one can numerically encode the categories with equal spacing and permutate the encoding when adding the nominal covariate as a dimension to a tessellation, this allows all class interactions to be considered.

2.2 Sum-of-tessellations

Our sum-of-tessellations model consists of a predetermined fixed number m of tessellations, a hyperparameter in our proposed method. T_j denotes the structure of the j^{th} tessellation, that is, which covariates are dimensions in the tessellation and the coordinates of the centers of the cells. The corresponding values for each cell are given by $\boldsymbol{M}_j = \{\mu_{1j}, \ldots, \mu_{b_j j}\}$, where b_j is the number of cells in tessellation T_j. The output value for a sample $\boldsymbol{x}$ of tessellation T_j is given by the function $g(\boldsymbol{x} \mid T_j, \boldsymbol{M}_j)$. If we consider the case in which the model consists of only a single tessellation, then we have

$$Y = g(\boldsymbol{x} \mid T_1, \boldsymbol{M}_1) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2), \ (1)$$

where $g(\boldsymbol{x} \mid T_1, \boldsymbol{M}_1)$ is the average of all the samples in the given cell that $\boldsymbol{x}$ is in. In the AddiVortes model, an estimation of $\mathrm{E}[Y \mid \boldsymbol{x}, \boldsymbol{y}_{train}, \boldsymbol{X}_{train}]$ is given by the sum of all the outputs of the tessellations that $\boldsymbol{x}$ corresponds too, that is

$$Y = \sum_{j=1}^{m} g(\boldsymbol{x} \mid T_j, \boldsymbol{M}_j) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2).$$

If $m = 1$, then this case is the same as case 1, but, if $m > 1$, then each tessellation captures a fraction of $\mathrm{E}[Y \mid \boldsymbol{x}, \boldsymbol{y}_{train}, \boldsymbol{X}_{train}]$. In single-dimension tessellations, the μ_{ij} represent main effects since $g(\boldsymbol{x} \mid T_j, \boldsymbol{M}_j)$ only depends on a single covariate but will represent interaction effects when the tessellations are multi-dimensional. Thus, AddiVortes can capture both the main effects and interaction effects in the model. The other advantage of using an additive approach is that we will be considering several tessellations with smaller dimensions and centers and there exists algorithms that efficiently find the associated centers of samples for this case. We use the FNN: Fast Nearest Neighbor Search Algorithms and Applications CRAN package from Alina Beygelzimer et al. (2023) to implement our algorithm.

2.3 Regularization priors

Priors are specified for the parameters of the sum-of-tessellations model, specifically for $(T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m)$, and σ. These priors are strategically chosen to favor less complex configurations with fewer dimensions and cells. This regularization controls the influence of individual tessellation effects, preventing any one of them from becoming dominant. Without these regularization priors, complex tessellations with a large number of dimensions and centers would cause over-fitting and limit the advantages of the additive model in both approximation and computation.

To simplify the implementation of AddiVortes, we propose default specifications for the priors, which have proven effective in numerous examples (as detailed in Sections 4 and 5). We achieve this by reducing the prior formulation problem to a few interpretable hyperparameters governing priors on T_j, $\boldsymbol{M}_j$ and σ. Our recommended defaults are obtained by leveraging the observed variation in *Y* to estimate reasonable hyperparameter values. As an alternative strategy, one can specify a range of plausible hyperparameter values using the considerations provided and then utilize cross-validation to select the most suitable ones. However, it is important to note that this approach will require more computational resources that may not justify the improved accuracy.

2.3.1 Prior independence and symmetry

The specification of the regularization priors is simplified if we apply independence restrictions, similar to one found in Chipman et al. (2010), such that,

$$\pi\left((T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \sigma\right) \quad = \prod_{j=1}^{m} \left[\pi(T_j, \boldsymbol{M}_j)\right] \pi(\sigma)$$

$$= \prod_{j=1}^{m} \left[\pi\left(\boldsymbol{M}_j \mid T_j\right) \pi(T_j)\right] \pi(\sigma)$$

and

$$\pi(\boldsymbol{M}_j \mid T_j) = \prod_{i=1}^{b_j} \pi\left(\mu_{ij} \mid T_j\right),$$

where $\mu_{ij} \in \boldsymbol{M}_j$. Here, we have restricted the prior distributions so that the tessellations are independent of each other and of σ *a priori* and the output values in a given tessellation are also independent of each other. Note, however, that the acceptance probability of a new tessellation in the MCMC algorithm (as described in Section 3) is dependent on the other tessellations in the model and induces dependence *a posteriori*.

By imposing independence here, we greatly simplify the process of defining prior distributions. Specifically, we only need to specify forms for $\pi(T_j)$, $\pi(\mu_{ij} \mid T_j)$, and $\pi(\sigma)$, rather than dealing with complex joint distributions across all tessellations. This simplification is further enhanced by using identical prior forms for all tessellations' structure $\pi(T_j)$ and all parameters within each tessellation $\pi(\mu_{ij} \mid T_j)$.

In practice, we adopt similar prior distributions to the ones proposed by Chipman et al. (1998) for the Bayesian CART (Classification and Regression Trees) model. In the subsequent subsections, we will observe that these prior forms offer computational efficiency and are defined by a concise set of interpretable hyperparameters, enhancing their practical value.

2.3.2 T_j prior

The prior for the tessellation structure T_j is specified by multiple factors:

1. the number of covariates considered in T_j,

2. the number of centers in T_j,

3. the covariates that are included in T_j and

4. the coordinates of the centers.

We assign the probability of the number of covariates a (shifted) binomial distribution, $d_j - 1 \sim \text{Binomial}(p-1, \frac{\omega}{p})$ where d_j is the number of covariates included in tessellation *j*, and *p* is the total number of covariates. The probability density for the number of centers in the model is a (shifted) Poisson distribution $b_j - 1 \sim \text{Poisson}(\lambda_c)$, where b_j is the number of cells in tessellation T_j and λ_c is the rate parameter. The probability that a covariate is chosen as a dimension in a tessellation follows a (discrete) uniform distribution over the remaining covariates that are not already in the tessellation.

For all tessellations, we assign the probability of the coordinates of the centers a normal distribution: $\mathcal{N}(0, \sigma_c^2)$. We set the mean of the normal distribution as 0 as we have performed a linear transformation to the covariates so that the maximum and minimum values are at 0.5 and -0.5 and in many cases the mean of the transformed covariates will be near zero. Most centers, that will partition the data such that some observations fall within the new cell created by the new center, will have coordinates between -0.5 and 0.5, the maximum and minimum values of the covariates after transformation. Thus, we choose a value for σ_c so that the probability that the majority of the centers have coordinates within the interval $(-0.5, 0.5)$, that is between the maximum and minimum values of the observed covariate measurement. However, there are cases that the coordinates of the centers will fall outside this range, thus the normal distribution is useful as it assigns a lower probability for coordinates the higher the absolute value of the coordinates.

2.3.3 $\mu_{ij} \mid T_j$ prior

We handle the parameters μ_{ij} using a conjugate normal distribution $N(\mu_\mu, \sigma_\mu^2)$ similar to one used in Chipman et al. (2010), which confers significant computational advantages as we can easily marginalize μ_{ij}. One of our objectives is to estimate $\mathrm{E}[Y \mid \boldsymbol{x}, \boldsymbol{y}_{train}, \boldsymbol{X}_{train}]$, which equals the sum of the output values over all tessellations. Since the μ_{ij} are independent and identically distributed, the induced prior on $\mathrm{E}[Y \mid \boldsymbol{x}, \boldsymbol{y}_{train}, \boldsymbol{X}_{train}]$ follows a normal distribution $N(m\mu_\mu, m\sigma_\mu^2)$.

To ensure that $\mathrm{E}[Y \mid \boldsymbol{x}, \boldsymbol{y}_{train}, \boldsymbol{X}_{train}]$ predominantly lies within the observed range $(y_{\min}, y_{\max})$, where $y_{\min}$ and $y_{\max}$ represent the minimum and maximum observed values of the output *Y*, we again choose appropriate hyperparameters. Specifically, we set the hyperparameters such that $m\mu_\mu - k\sqrt{m}\sigma_\mu = -0.5$ and $m\mu_\mu + k\sqrt{m}\sigma_\mu = 0.5$, where *k* is a preselected hyperparameter determining the prior probability of $\mathrm{E}[Y \mid \boldsymbol{x}, \boldsymbol{y}_{train}, \boldsymbol{X}_{train}]$ being within the interval $(y_{\min}, y_{\max})$.

Given the prior centered at 0 ($\mu_\mu = 0$), the prior distribution for μ_{ij} is expressed as:

$$\mu_{ij} \sim N(0, \sigma_\mu^2), \quad \text{where} \quad \sigma_\mu = \frac{0.5}{k\sqrt{m}}.$$

This prior encourages the tessellation parameters μ_{ij} to shrink towards zero, effectively constraining the influence of individual tessellation components and keeping them relatively small. As we increase *k* and/or the number of tessellation components *m*, this prior becomes more restrictive, leading to stronger shrinkage of the μ_{ij}.

2.3.4 σ prior

For σ, a conjugate prior is employed: a natural choice for this prior is the inverse chi-square distribution $\sigma^2 \propto \nu\lambda / \chi_\nu^2$. To guide the calibration of hyperparameters ν and λ, a data-informed approach is adopted where we assign a significant probability to a range of credible σ values while maintaining a balance between concentration and dispersion within the distribution. The objective is to adjust the prior's degrees of freedom ν and scale parameter λ using a "rough data-based overestimate" $\hat{\sigma}$ of σ.

Two plausible choices for $\hat{\sigma}$ are considered. The first is the "naive" option, wherein $\hat{\sigma}$ corresponds to the sample standard deviation of the transformed training response values. The second is the "linear model" choice, where $\hat{\sigma}$ is based on the residual standard deviation from a least-squares linear regression of scaled *Y* on the scaled $\boldsymbol{X}$ variables. Subsequently, a value for ν within the range of 3 to 10 is selected to shape the prior appropriately. Additionally, a suitable λ value is determined such that the q^{th} quantile of the prior on σ aligns with $\hat{\sigma}$, that is $Pr(\sigma < \hat{\sigma}) = q$. A suitable range for *q* is values between 0.7 and 0.99.

2.4 Choice of *m*

A fixed number of tessellations are used, and an iterative backfitting algorithm is employed to cycle through these tessellations. Determining an appropriate value for the number of tessellations *m* presents a challenge. Two common strategies to address this challenge are: (1) to treat *m* as an unknown parameter and assign a prior to it and (2) to use cross-validation to select the 'best' value for *m* from a range of reasonable choices. However, both of these strategies incur significant computational costs. Typically, as illustrated in Section 5, increasing *m* from one leads to considerable enhancements in predictive performance until a point where it levels off. Beyond this point, for large *m* values, performance slowly degrades, because each tessellation contributes such a small fraction of the expected outcome and there is uncertainty within each tessellation.

To avoid computational overhead, a practical approach is to start with a default value of $m = 200$. In numerous scenarios, this default value has demonstrated excellent predictive performance and this is illustrated in Section 4 and 5. However, other considerations might come into play, particularly when using AddiVortes for variable selection, where the choice of *m* could be influenced by additional factors.

3 MCMC backfitting algorithm

The aim of AddiVortes is to extract the posterior distribution of all unknown parameters in the sum-of-tessellations model,

$$\pi((T_1, \boldsymbol{M}_1), \dots, (T_m, \boldsymbol{M}_m), \sigma \mid \boldsymbol{y}, \boldsymbol{X}), \quad (2)$$

by utilizing the MCMC backfitting algorithm. Note to simplify notation, in this chapter, we let $\boldsymbol{y}$ and $\boldsymbol{X}$ denote the training data for the output variables and the covariates, respectively. The Gibbs sampler involves *m* successive draws of $(T_j, \boldsymbol{M}_j)$ conditionally on $(\{T_{j'}, \boldsymbol{M}_{j'}\}_{j \neq j'}, \sigma, \boldsymbol{y}, \boldsymbol{X})$ for $j = 1, \dots, m$, followed by a draw of σ from its full conditional distribution. Hastie and Tibshirani (2000) previously explored a similar application of the Gibbs sampler for posterior sampling in additive and generalized additive models, with σ held fixed. They demonstrated that this approach is a stochastic generalization of the backfitting algorithm used in such models.

The draw of σ from the full conditional can simply be achieved by sampling from an inverse gamma distribution,

$$\pi\left(\sigma^2 \mid (T_1, \boldsymbol{M}_1), \dots, (T_m, \boldsymbol{M}_m), \boldsymbol{y}, \boldsymbol{X}\right) \propto \pi(\boldsymbol{y} \mid (T_1, \boldsymbol{M}_1), \dots, (T_m, \boldsymbol{M}_m), \sigma^2)\pi(\sigma^2)$$

$$\propto \mathrm{IG}\left(\frac{\nu + n}{2}, \frac{\nu\lambda + \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{m} g(x_i \mid T_j, \boldsymbol{M}_j) \right)^2}{2} \right).$$

It is more difficult to sequentially sample $(T_j, \boldsymbol{M}_j) \mid (\{T_{j'}, \boldsymbol{M}_{j'}\}_{j \neq j'}, \sigma, \boldsymbol{y})$ for all *m* tessellations. It is important to note that the conditional probability $\pi(T_j, \boldsymbol{M}_j \mid \{T_{j'}, \boldsymbol{M}_{j'}\}_{j \neq j'}, \sigma, \boldsymbol{y})$ relies solely on $(\{T_{j'}, \boldsymbol{M}_{j'}\}_{j \neq j'}, \boldsymbol{y}, \boldsymbol{X})$ through the expression

$$\boldsymbol{R}_j = \boldsymbol{y} - \sum_{k \neq j} g(\boldsymbol{x} \mid T_k, \boldsymbol{M}_k),$$

where $\boldsymbol{R}_j$ represents the *n*-vector of partial residuals derived from a fitting process that excludes the j^{th} tessellation. Thus, the *m* draws of $(T_j, \boldsymbol{M}_j)$ given $(T_{(j)}, \boldsymbol{M}_{(j)}, \sigma, \boldsymbol{y}, \boldsymbol{X})$ are equivalent to *m* draws from

$$(T_j, \boldsymbol{M}_j) \mid \boldsymbol{R}_j, \sigma, \quad (3)$$

$j = 1, \ldots, m$. Now, (3) is formally equivalent to the posterior of the single tessellation model $\boldsymbol{R}_j = g(x; T_j, \boldsymbol{M}_j) + \epsilon$ where $\boldsymbol{R}_j$ plays the role of the data $\boldsymbol{y}$. Since we have used a conjugate prior for $\boldsymbol{M}_j$,

$$\pi(Tj \mid Rj, \sigma) \propto \pi(Tj) \int \pi(\boldsymbol{R}_j \mid \boldsymbol{M}_j, T_j, \sigma) \pi(\boldsymbol{M}_j \mid T_j, \sigma) d\boldsymbol{M}_j \quad (4)$$

can be obtained in closed form up to a normalizing constant. This allows us to carry out each draw from (4) in two successive steps as

$$\begin{aligned} &T_j \mid \boldsymbol{R}_j, \sigma, \\ &\boldsymbol{M}_j \mid T_j, \boldsymbol{R}_j, \sigma. \end{aligned} \quad (5)$$

The draw of T_j in (5) can be obtained using the Metropolis–Hastings (MH) algorithm similar to the one proposed by Chipman et al. (1998). Six moves are suggested to propose a new tessellation based on the current tessellation, each with its associated proposal probability:

• adding/removing a center (0.2 each),

• adding/removing a covariate (0.2 each),

• swapping a covariate (0.1) or

• changing the position of a center (0.1).

To incorporate a new center, we simply sample coordinates (from a normal distribution described in Section 2.3.2) for each dimension in the tessellation. To eliminate a center, we uniformly sample one and remove it. Adding a dimension involves uniformly selecting a new dimension from ones not in the tessellation and sampling coordinates for each center, while removing a dimension entails uniformly sampling a dimension in the tessellation and deleting it by removing all its coordinates. Changing a center involves uniformly sampling one and sampling new coordinates. Swapping a covariate requires uniformly sampling a dimension to delete, then randomly selecting a covariate not in the tessellation and sampling coordinates for the new dimension for each center.

Note, adjustments to the acceptance probability need to be made to take into account when certain moves cannot occur, for example a center cannot be removed if there is only one center in the tessellation. These adjustments are described in Appendix B.

By integrating out $\boldsymbol{M}_j$ in (4), we avoid the complexities of reversible jumps between continuous spaces of varying dimensions Green (1995).

Subsequently, the draw of $\boldsymbol{M}_j$ in Equation (5) involves independent draws of μ_{ij} from a normal distribution. The draw of $\boldsymbol{M}_j$ enables the calculation of the subsequent residual R_{j+1}, which is essential for the next draw of T_j.

We initiate the chain with *m* simple single-center one-dimensional tessellations, and then we repeat iterations for a burn-in period until satisfactory convergence is achieved. We usually choose a relatively small burn-in period of 200 iterations since convergence is fast, as illustrated in Section 5.

Algorithm 1 Bayesian backfitting MCMC for posterior inference in AddiVortes

Data: Training data $(\boldsymbol{X}_{train}, \boldsymbol{y}_{train})$, AddiVortes hyperparameters $(m, \nu, q, k, \sigma_c, \omega, \lambda_c)$

```
Find $\lambda$ ;               // Using naive or linear method
for $j = 1$ to m do
    Initialize tessellation $T_j$ ;      // Create m single center tessellations
    Assign $\mu_{1j}$ ;                  // $\mu_{1j} = \text{mean}(\boldsymbol{y}_{train})$
for $i = 1$ to number of iterations do
    Sample $\sigma^2$ ;                  // Sample from inverse gamma distribution
    for $j = 1$ to m do
        Compute residuals $R_j$ ;        // $R_{ij} = y_i - \sum_{k \neq j} g(x_i \mid T_k, \boldsymbol{M}_k)$
        Propose new tessellation;
        Accept or reject tessellation;
        Sample $\mu_{ij}$ ;              // Sample from conjugate normal distribution
```

In Section 5, we see that the backfitting MCMC algorithm performs well, even for challenging cases, and repeated use of the algorithm yield similar results consistently. As the results are consistent, we typically run one long chain with AddiVortes instead of using multiple starts. The backfitting MCMC algorithm for the sum-of-tessellations model shows significant advantages in terms of mixing and flexibility, making it a powerful tool for various complex modeling tasks.

3.1 Inference for the model parameters

The backfitting algorithm described in the previous section is ergodic, producing a sequence of draws $(T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \sigma$ that converges (in distribution) to the posterior

$$\pi((T_1, \boldsymbol{M}_1), \ldots, (T_m, \boldsymbol{M}_m), \sigma \mid \boldsymbol{y}, X).$$

The corresponding sequence of sum-of-tessellations functions is given by

$$f(\cdot) = \sum_{j=1}^{m} g(\cdot; T_j^*, M_j^*) \quad (6)$$

where $(T_1^*, M_1^*), \ldots, (T_m^*, M_m^*)$ are the draws from the posterior. This sequence of functions is thus converging to the posterior distribution $\pi(f \mid \boldsymbol{y})$ on the “true” function $f(\cdot)$. By running the algorithm for a sufficient number of iterations after a suitable burn-in period, we obtain a sample $f_1^*, \ldots, f_K^*$ that can be treated as an approximate, dependent sample of size K from $\pi(f \mid \boldsymbol{y})$. To estimate $f(\boldsymbol{x})$, we can take the average or median of the posterior samples.

A $(1-\alpha)\%$ posterior interval for $f(\boldsymbol{x})$ is the interval between the upper and lower $\alpha/2$ quantiles but this a “Naive” approach as it doesn’t consider the noise ϵ. Alternatively, “true” prediction intervals which includes the affect of the noise ϵ and its estimation error can be obtained by adding the noise to each posterior sample for each observation and taking the upper and lower $\alpha/2$ quantiles. These uncertainty intervals behave sensibly, widening at $\boldsymbol{x}$ values far from the data.

In some cases, thinning may be appropriate. Thinning the chain, explained in Owen (2017), is a technique used to reduce auto-correlation in the samples by only retaining every k^{th} sample from the sequence. This is done to ensure that the retained samples are more independent of each other, which can improve the accuracy of parameter estimates and the efficiency of the algorithm.

4 Performance on regression data sets

We conducted a comprehensive comparison of the predictive performance of AddiVortes against several competing algorithms across a range of data sets obtained from Loh et al. (2007). Due to availability of data sets we completed our analysis on a a fraction of data sets listed and the ones used are given in Table 1. These datasets exhibit varying sample sizes, ranging from 96 to 4,177, with each dataset comprising an output variable and between 4 and 21 covariates. We applied the one-hot encoding method to handle categorical covariates since all the data sets we considered had fewer than five categories for all covariates.

For each data set, we created 20 independent train/test splits, with 5/6 of the data allocated to the training set. Two versions of AddiVortes were considered: AddiVortes-CV, where prior hyperparameters $(m, \nu, q, k, \sigma_c, \omega, \lambda_c)$ were chosen via five-fold cross-validation, and AddiVortes-default, utilizing predetermined hyperparameters based analysis on many simulated and real world data sets, $(m, \nu, q, k, \sigma_c, \omega, \lambda_c) = (200, 6, 0.85, 3, 0.8, 3, 25)$, as detailed in the supplementary material. A burn-in of 200 iterations was chosen, demonstrating sufficient convergence in many cases.

As competitors, we evaluated four black-box methods: random forests (Breiman, 2001, implemented as `randomforest` in R), gradient boosting (Freund and Schapire, 1997, implemented as `gbm` in R), BART (Chipman et al., 2010, implemented as `bart` from the BayesTree R package) and SoftBART (Linero and Yang, 2018, implemented as `SoftBart` in R). These models were selected for their robust multivariate regression capabilities, interpretability and comparability.

All methods, except AddiVortes-default, underwent five-fold cross-validation to select hyperparameters and the considered values for each method are provided in Table 2. In particular, for AddiVortes-CV, we considered the following cross validation ranges:

- $k = 1, 3$, reflecting moderate to heavy shrinkage for the μ prior hyperparameter,
- $\sigma_\mu = 0.255, 0.8$, considering low to high variation in the coordinates of centers,
- $\lambda_c = 5, 25$, evaluating lower and higher number of centers in each tessellation, and
- $m = 50, 200$, for the number of trees,

a total of $2^4 = 16$ potential choices for $(k, \sigma_\mu, \lambda_c, m)$. All other hyperparameters take their default value as they are less influential on performance and considering more values would increase computational time.

To facilitate performance comparisons across data sets, we employed the relative RMSE (RRMSE), defined as the RMSE divided by the minimum RMSE obtained by any method for each test/train split. This normalization allows for meaningful comparisons irrespective of location and scale transformations of the response variables. Boxplots of the RRMSE values for each method across test/train splits are depicted in Figure 2 and the (50%, 75%) RRMSE quantiles are provided in Table 3.

Note, we didn't included gradient boosting in the box-plot as the RRMSE values was much higher then the competitors included. We also removed all RRMSE values greater then 1.5 so that the box-plot gave a better comparison between competing methods. No RRMSE value was removed for AddiVortes-CV or BART, only one value for AddiVortes-default, four values for SoftBART and nine values for random forests.

While the relative performance in Figure 2 exhibits variation across different data sets, it is evident from the RRMSE distribution that AddiVortes-CV consistently achieved smaller RMSE values more frequently than its competitors. Notably, AddiVortes-default exhibited impressive overall performance. This is particularly noteworthy given that random forests, SoftBART and BART relied on cross-validation for hyperparameter tuning, while AddiVortes-default excelled without the need for such specifications.

5 Illustrating AddiVortes on simulated data

We use simulated data to evaluate the performance of AddiVortes against known values and consider the set up originally explored in Friedman (1991). We generate data by simulating the values of $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$, with $x_1, x_2, \ldots, x_p$ being independently drawn from the standard uniform distribution and the response variable *Y* given by

$$Y = f(\boldsymbol{x}) + \epsilon = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon, \quad (7)$$

where $\epsilon \sim \mathcal{N}(0,1)$. Notably, *Y* is solely dependent on $x_1, \ldots, x_5$, so the predictors $x_6, \ldots, x_p$ are inactive variables. The introduction of these extraneous variables injects a layer of complexity.

5.1 Illustrating AddiVortes' abilities

To help compare the AddiVortes model with BART, we have used examples with similar iterations to the ones used in Chipman et al. (2010). To begin, we highlight the fundamental aspects of AddiVortes by employing a single simulated data set from the Friedman function, with $p = 10$ predictors and $n = 150$ observations. To facilitate our illustration, we apply the default values $(m, \nu, q, k, \sigma_c, \omega, \lambda_c) = (200, 6, 0.85, 3, 0.8, 3, 25)$ and we generate 1,800 MCMC posterior samples f^*, discarding the initial 200 iterations as burn-in.

For each unique $\boldsymbol{x}$ value, we compute posterior mean estimates $\hat{f}(\boldsymbol{x})$ by averaging the 1,800 posterior samples $f^*(\boldsymbol{x})$, and we ascertain the endpoints of "Naive" 90% posterior intervals for each $f(\boldsymbol{x})$ by identifying the 5% and 95% quantiles of the f^* values. Figure 3 (a) illustrates the predicted output values $\hat{f}(\boldsymbol{x})$ vs the true values $f(\boldsymbol{x})$ for the $n = 150$ in-sample $\boldsymbol{x}$ values that generated the corresponding *Y* values through Equation (7) and the vertical lines denote the 90% posterior intervals for $f(\boldsymbol{x})$. Here, we see that the $\hat{f}(\boldsymbol{x})$ values are near the true $f(\boldsymbol{x})$ values, while the true values are within the intervals. Figure 3 (b) extends this to 150 randomly selected out-of-sample $\boldsymbol{x}$ values. Although the correlation between $\hat{f}(\boldsymbol{x})$ and $f(\boldsymbol{x})$ is slightly weaker, accompanied by slightly wider intervals, AddiVortes performs considerably well with a small number of training samples.

Note that the 90% posterior intervals may not precisely match 90% frequentist coverage and, in this case, for the training data the intervals cover 92% the true values and 98% of the true values for the testing data. In scenarios involving real-world data where *f* remains unknown, bootstrap and cross-validation methodologies could provide insights into interval coverage. Notably, for extreme $\boldsymbol{x}$ values, the influence of the prior could entail a more pronounced shrinkage towards zero, which leads to reduced coverage frequencies.

Figure 3 (c) shows the draw of σ and the horizontal line symbolizes the true value of $\sigma = 1$. This illustrates the Markov chain's quick convergence to equilibrium and the σ draws varying around the true value $\sigma = 1$ suggests the model is not over-fitting.

Figure 4 shows for the average number of dimensions (left) and average number of centers (right) for the simulation above, with the default hyperparameters used and 2000 iterations. There is quick convergence to approximately 2.4 and 3.5, for the average number of dimensions and centers, respectively. However, the graphs do suggest that the tessellations are still exploring the space since these averages are still oscillating around these values.

AddiVortes estimates the partial dependence function [Friedman (2001)], which summarizes the individual x_i 's marginal effects on *Y*. Figure 5 provides point estimates of the partial dependence functions for $x_1, \ldots, x_{10}$ derived from the 5,000 posterior samples. This illustrates the impact of $x_1, \ldots, x_{10}$ on *Y*, clearly there is significant marginal effects for $x_1, \ldots, x_5$ and almost constant marginal effects for $x_6, \ldots, x_{10}$.

In addition to prediction, AddiVortes can be used as a variable selection tool by identifying the variables that are most frequently included in accepted tessellations. Figure 6 illustrates the percentage of each variable used in accepted tessellations for each covariate $x_1, \ldots, x_{10}$, across 5,000 posterior draws f^*, across a range of *m* values: 10, 20, 50, 100, and 200. We use a sample of $n = 500$ simulated observations of the Friedman function (7) with $p = 10$, matching the graph used in the BART paper. When the number of tessellations *m* is lower, the ensemble of sum-of-tessellations models progressively includes only the variables—namely $x_1, \ldots, x_5$—causing *Y*'s variation. Without invoking any presumptions or knowledge of the actual functional form of *f* in Equation (7), AddiVortes adeptly identifies the subset of variables that underpin the dependency of *f*.

AddiVortes is robust to changes in the hyperparameters. To illustrate this, we use 150 samples of the Friedman equation with $p = 10$ and we calculate the RMSE of 150 out-of-sample values and run AddiVortes 100 times for 3 cases: aggressive, default and conservative. For the aggressive hyperparameters, we chose $(\nu, q, k, \sigma_c, \omega, \lambda_c) = (10, 0.75, 3, 0.8, 1, 5)$ where there is high shrinkage towards zero and probability of lower dimension tessellations with less centers. Whereas, for the conservative setting hyperparameters $(\nu, q, k, \sigma_c, \omega, \lambda_c) = (6, 0.95, 1, 1.5, 3, 25)$, there is lower shrinkage towards zero and higher probability for higher dimensional tessellations with a higher number of centers. The default setting, $(\nu, q, k, \sigma_c, \omega, \lambda_c) = (6, 0.85, 3, 0.8, 3, 25)$, is between the aggressive and conservative setting. Figure 7 shows the RMSE for the 3 hyperparameter settings for *m* increasing from 1 to 500, and the 90% intervals for each case. We see the RMSE values are close for all settings for any number of tessellations, with the aggressive setting performing worse for smaller values of *m*. The graph also illustrates a trend in RMSE values with varying numbers of tessellations. Initially, there is a noticeable improvement in RMSE values as we increase the number of tessellations from 1 to 50. However, as the number of tessellations continues to rise beyond 200, there is a gradual degradation in RMSE values. This is probably due to the uncertainty in individual tessellations. Notably, when the number of tessellations exceeds 100, the 90% intervals become much smaller. This implies that multiple algorithm runs consistently produce similar results and it is best to run one long chain instead of multiple smaller chains.

Next, we consider higher-dimensional data, for the function *f* in Equation (7), with its dependency on only five variables ($x_1, \ldots, x_5$). Building upon our earlier exploration of AddiVortes in Friedman's setup with $p = 10$ and $n = 150$ observations, we include higher *p* values. In the simulations, AddiVortes' ability to find low-dimensional relationships when there is a high number of inactive covariates in the data is highlighted. We replicate the illustrations featured in Figure 3, now with $p = 20$, 100, and 1000, all while keeping a small number of observations.

For $p = 20$, we used the hyperparameters values similar to the default setting but with $m = 50$ instead of 200. For $p = 100$ and $p = 1000$, the hyperparameters used were $(m, \nu, q, k, \sigma_c, \omega, \lambda_c) = (50, 6, 0.99, 3, 0.2, 3, 25)$ and we use the naive estimate $\hat{\sigma}$ (sample standard deviation of *Y*) instead of the least squares estimate to anchor the *q*th prior quantile, adeptly accommodating the problems arising when $p \geq n$.

Figure 8 highlights the performance of AddiVortes to for in-sample and out-of-sample data when the *p* value is increased and the coverage of the 90% posterior intervals are give in Table 4. The in-sample approximations and their accompanying 90% intervals for $f(\boldsymbol{x})$ are impressively accurate across all *p* values. In the out-of-sample scenario, with larger *p*, the estimation of extreme $f(\boldsymbol{x})$ shifts towards the mean. The AddiVortes algorithm performs extremely well with a small number of samples training the model, even with a very high number of inactive covariates.

In the third column of Figure 8, an illustration of the MCMC progression of σ estimations with a solid line at $\sigma = 1$, the known variance of the data. This σ draws frequently cross the line for $\sigma = 1$ but with larger *p*, it increasingly tends to stray back toward larger values, a reflection of increasing uncertainty.

An appealing feature of AddiVortes is its robustness to being misled by inactive covariates. To explore this, we simulated $n = 150$ observations with $f \equiv 0$, for $p = 10$, 100, and 1000. Using these settings, AddiVortes indicated that *f* intervals—both in-sample and out-of-sample—centered around 0 for $p = 10$ and $p = 100$, indicating no relationship. For $p = 1000$, where data provides limited insights, some in-sample intervals moved away from 0 due to the prior's influence. However, out-of-sample 90% posterior intervals consistently included zero.

5.2 Evaluating Out-of-Sample Performance across Competing Approaches

For our simulated study to gauge the effectiveness of AddiVortes within the Friedman setup, we compare random forests, BART, gradient boosting and SoftBART.We estimate the function *f* with $n = 200$ observations with $p = 10$. To conduct this experiment, for AddiVortes model, we performed 1,000 MCMC iterations after discarding an initial 200 draws as burn-in. For hyperparameter specification, we employed 5-fold cross-validation with values from Table 2.

We simulated 100 data sets, each comprising $n = 200$ observations. Given that the data generating process is known, we did not need to simulate a test set; instead, for each method's estimate $\hat{f}$ derived from each data set, 800 independent $\boldsymbol{x}$ values were randomly selected to assess the fit using the RMSE, calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{800}\sum_{i=1}^{800}(\hat{f}(x_i) - f(x_i))^2}.$$

Thus, we obtained 100 RMSE values for each method.

Figure 9 illustrates the outcomes using a box plot (note that the box plot in Figure 9 represents RMSE values and not relative RMSE values as in Figure 2). As expected, SoftBART performs particularly well against its competitors because it uses a sparsity inducing prior for selecting variables to use in the trees introduced by Linero (2016), a modification that can easily be applied to AddiVortes. AddiVortes performs slightly better then BART but both perform much better then gradient boosting and random forests. This shows that the AddiVortes can find relevant variables for predicting outcomes and can out-perform leading black-box methods. Once further modifications and optimizations are applied to AddiVortes it is likely to perform as well or even better then SoftBART.

6 Computational Expense

The AddiVortes Algorithm is currently not optimized to be computationally efficient when running. For example, we have coded the algorithm in R studio which is a slower programming language then many competitors. Despite using the efficient FNN: Fast Nearest Neighbor Search Algorithms and Applications CRAN package that uses C++, AddiVortes can be programmed to run faster than the current version.

The advantage of using an additive approach means that we can regularize the tessellation structures thereby limiting the number of centers and dimensions. Thus, when the number of covariates increases it does not have a significant effect on the computational time and there exists fast algorithms to find the centers of observations.

7 Discussion

The application of AddiVortes to diverse data sets and a simulation experiment highlights its attractive features. Notably, in terms of out-of-sample predictive RMSE performance, AddiVortes outperforms gradient boosting, BART and random forests. In simulation experiments, AddiVortes consistently yields reliable posterior mean and interval estimates of the true regression function, along with correct marginal predictor effects. The performance is robust to hyperparameter specification even in higher-dimensional spaces. AddiVortes is a useful tool for variable selection as when running the algorithm with fewer tessellations the more influential covariates are included in tessellations.

Usually, models that use Voronoi tessellations, such as the one described in Pope et al. (2021), are very computationally expensive and can not be scaled for higher dimensional tasks. However, due to using an additive approach and regularization priors, we only consider low dimensional tessellations which are less computationally expensive to find output values as there exits algorithms that efficiently find the nearest center for an observation and increasing the number of covariates has negligible effect on computational time.

Currently, the AddiVortes algorithm has not been optimized in terms of coding and is currently computationally expensive when compared to other methods. For instance, in Kapelner and Bleich (2016), an R package called “bartMachine” was introduced that uses Java to optimize the BART algorithm and similar techniques could be used to reduce computational expense.

In this paper, we have exclusively looked at using the Euclidean metric space. However, further research could look into changing the metric used in the tessellations, for example using the Manhattan distance or using multiplicatively-weighted metrics. Some metrics would lead to the boundaries of the cells being curved and possibly partitioning the covariate space in a more suitable way for particular data sets.

AddiVortes can be adapted for binary classification by employing a probit link, similar to the one described in Zhang and Härdle (2010) to modify the BART algorithm, to predict the probability of activity. This adaptation incorporates data augmentation techniques outlined in Albert and Chib (1993). The approach can be further extended to handle classification problems, following a similar approach applied to BART described in Lee and Hwang (2024) for ordinal classification and Kindo et al. (2016) for nominal classification.

In the AddiVortes algorithm, tessellations partition the covariate space and output values are based on the cell the sample falls in. However, if we apply a probabilistic approach to which cell a sample falls into based on the distance to the center of its cell and the others then this would further smooth the output function of the algorithm. This approach is similar to to SoftBART presented in Linero and Yang (2018), where they modified BART such that the function was smoothed by adding a probabilistic aspect to each tree.

Expanding the model’s scope from regression, we note that BART has been successfully adapted for causal inference in Hill (2011). Similar adjustments to the AddiVortes algorithm can be made so that it can be used for causal inference applications. We believe that the AddiVortes algorithm will perform well in causal inference settings as tessellations flexibly partition the covariate space and determine interactions between variables well. For causal inference, further modifications to the algorithm can be made such as including the treatment application probability in the covariate space, similar to the method presented in Hahn et al. (2019), and applying shrinkage for data sets that are sparse, similar to the technique used in Caron et al. (2021).

8 Acknowledgments

We thank Dr Emmanuel Ogundimu for his insightful and constructive comments and suggestions that improved the paper.

9 Conflict of Interest statement

The authors report there are no competing interests to declare.

References

Albert, J. H. and Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. Journal of the American Statistical Association, 88(422):669–679.

Alina Beygelzimer, S. K., John Langford, S. A., and David Mount, S. L. (2023). Fast nearest neighbor search algorithms and applications.

Bock, M., Tyagi, A. K., Kreft, J.-U., and Alt, W. (2009). Generalized Voronoi tessellation as a model of two-dimensional cell tissue dynamics.

Breiman, L. (1996). Bagging predictors. Machine Learning, 24:123–140.

Breiman, L. (2001). Random forests. Machine Learning, 45(1):5–32.

Caron, A., Baio, G., and Manolopoulou, I. (2021). Shrinkage Bayesian causal forests for heterogeneous treatment effects estimation.

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

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

Dirichlet, G. L. (1850). Über die Reduction der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. Crelle’s Journal, 40:209–227.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. System Sci., 55:119–139. Second Annual European Conference on Computational Learning Theory (EuroCOLT ’95) (Barcelona, 1995).

Friedman, J. H. (1991). Multivariate adaptive regression splines. Ann. Statist., 19(1):1–141. With discussion and a rejoinder by the author.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Ann. Statist., 29(5):1189–1232.

Galceran, E. and Carreras, M. (2013). A survey on coverage path planning for robotics. Robotics and Autonomous Systems, 61(12):1258–1276.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika, 82(4):711–732.

Hahn, P. R., Murray, J. S., and Carvalho, C. (2019). Bayesian regression tree models for causal inference: regularization, confounding, and heterogeneous effects.

Hastie, T. and Tibshirani, R. (2000). Bayesian backfitting. Statist. Sci., 15(3):196–223.

Hill, J. L. (2011). Bayesian nonparametric modeling for causal inference. Journal of Computational and Graphical Statistics, 20(1):217–240.

Kapelner, A. and Bleich, J. (2016). bartmachine: Machine learning with Bayesian additive regression trees. Journal of Statistical Software, 70(4):1–40.

Kindo, B. P., Wang, H., and Peña, E. A. (2016). Mpbart - multinomial probit bayesian additive regression trees.

Kosaraju, N., Sankepally, S. R., and Mallikharjuna Rao, K. (2023). Categorical data: Need, encoding, selection of encoding method and its emergence in machine learning models—a practical review study on heart disease prediction dataset using Pearson correlation. In Saraswat, M., Chowdhury, C., Kumar Mandal, C., and Gandomi, A. H., editors, Proceedings of International Conference on Data Science and Applications, pages 369–382, Singapore. Springer Nature Singapore.

Lee, J. and Hwang, B. S. (2024). Ordered probit bayesian additive regression trees for ordinal data. Stat, 13(1):e643.

Li, H., Li, K., Kim, T., Zhang, A., and Ramanathan, M. (2012). Spatial modeling of bone microarchitecture. In Baskurt, A. M. and Sitnik, R., editors, Three-Dimensional Image Processing (3DIP) and Applications II, volume 8290, page 82900P. International Society for Optics and Photonics, SPIE.

Linero, A. (2016). Bayesian regression trees for high dimensional prediction and variable selection. Journal of the American Statistical Association, 113.

Linero, A. R. and Yang, Y. (2018). Bayesian Regression Tree Ensembles that Adapt to Smoothness and Sparsity. Journal of the Royal Statistical Society Series B: Statistical Methodology, 80(5):1087–1110.

Loh, W.-Y., Shih, Y.-S., and Chaudhuri, P. (2007). Visualizable and interpretable regression models with good prediction power. Iie Transactions, 39:565–579.

Musgrave, F. K., Kolb, C. E., and Mace, R. (1989). The synthesis and rendering of eroded fractal terrains. Proceedings of the 16th annual conference on Computer graphics and interactive techniques.

Owen, A. B. (2017). Statistically efficient thinning of a markov chain sampler.

Pope, C. A., Gosling, J. P., Barber, S., Johnson, J. S., Yamaguchi, T., Feingold, G., and Blackwell, P. G. (2021). Gaussian process modeling of heterogeneity and discontinuities using Voronoi tessellations. Technometrics, 63(1):53–63.

Shewchuk, J. R. (2002). Delaunay refinement algorithms for triangular mesh generation. Computational Geometry, 22(1):21–74. 16th ACM Symposium on Computational Geometry.

Voronoi, G. (1908). Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire. Sur quelques propriétés des formes quadratiques positives parfaites. Journal für die reine und angewandte Mathematik (Crelles Journal), 1908(133):97–102.

Zhang, J. L. and Härdle, W. K. (2010). The bayesian additive classification tree applied to credit risk modelling. Computational Statistics & Data Analysis, 54(5):1197–1205.

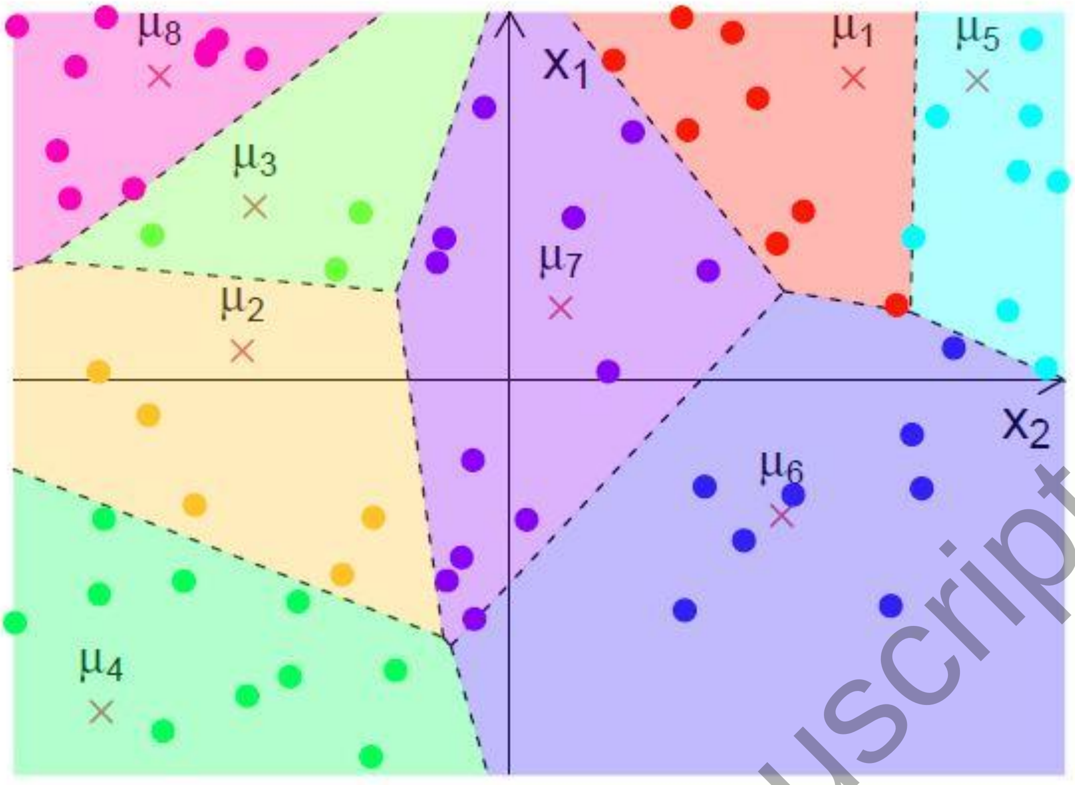

Figure 1: Example of predictive modeling using a two-dimensional Voronoi tessellation with centers at crosses, labeled with output values $(\mu_1, \ldots, \mu_8)$ associated to the given cell. Samples are represented by points with their output value corresponding to the cell they are in.

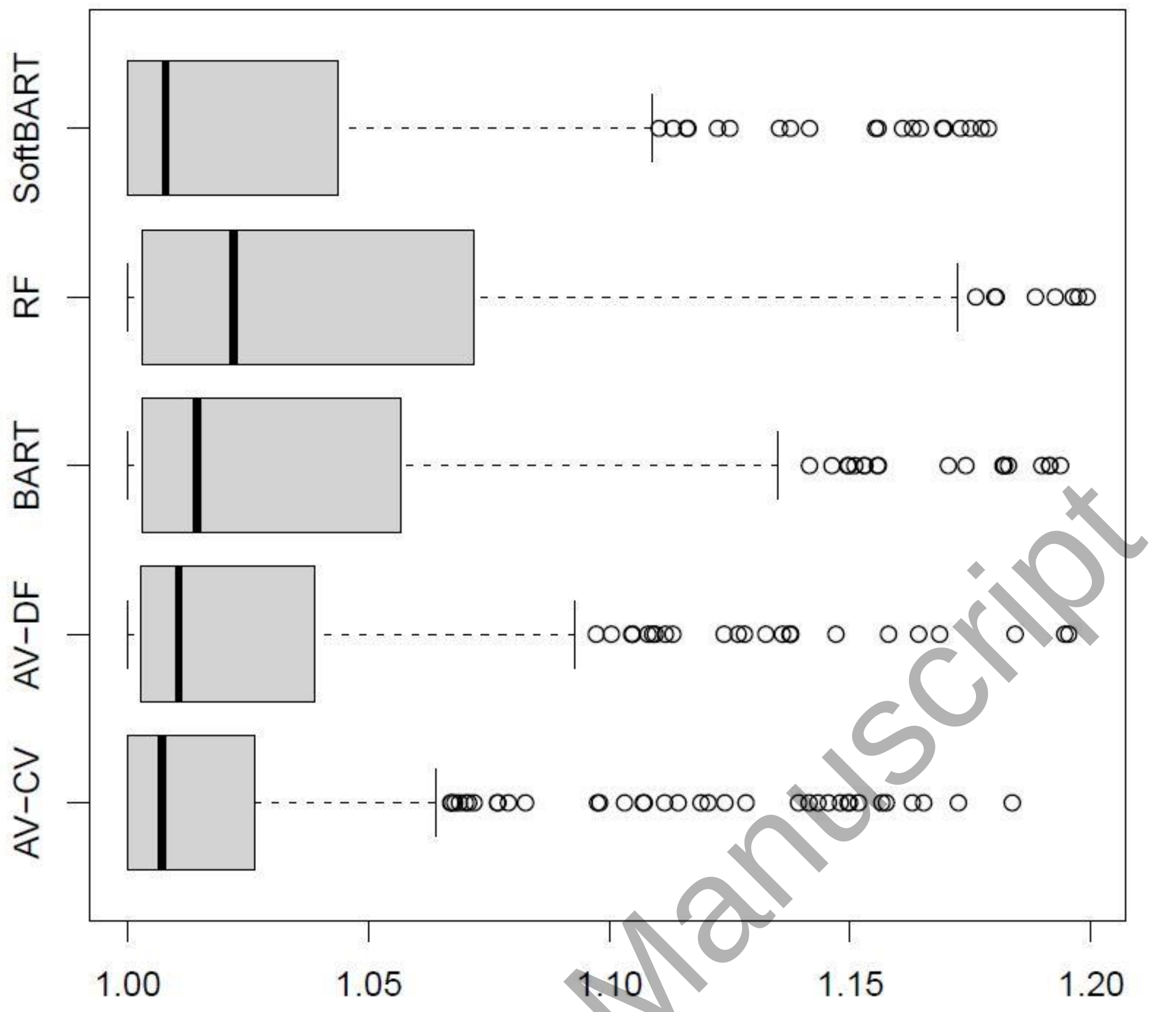

Figure 2: Boxplot of RRMSE for competing methods on 13 data sets.

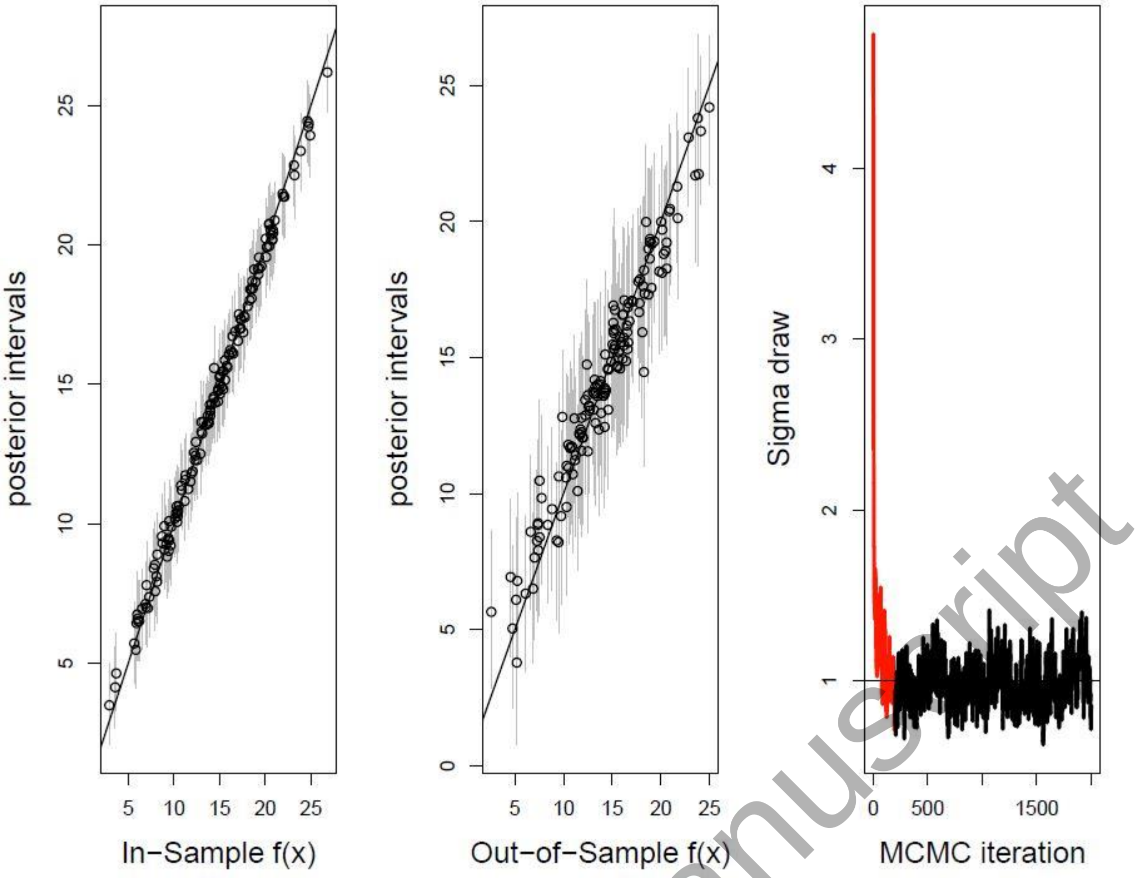

Figure 3: Inference about Friedman's $f(\boldsymbol{x})$ for p = 10 dimensions.

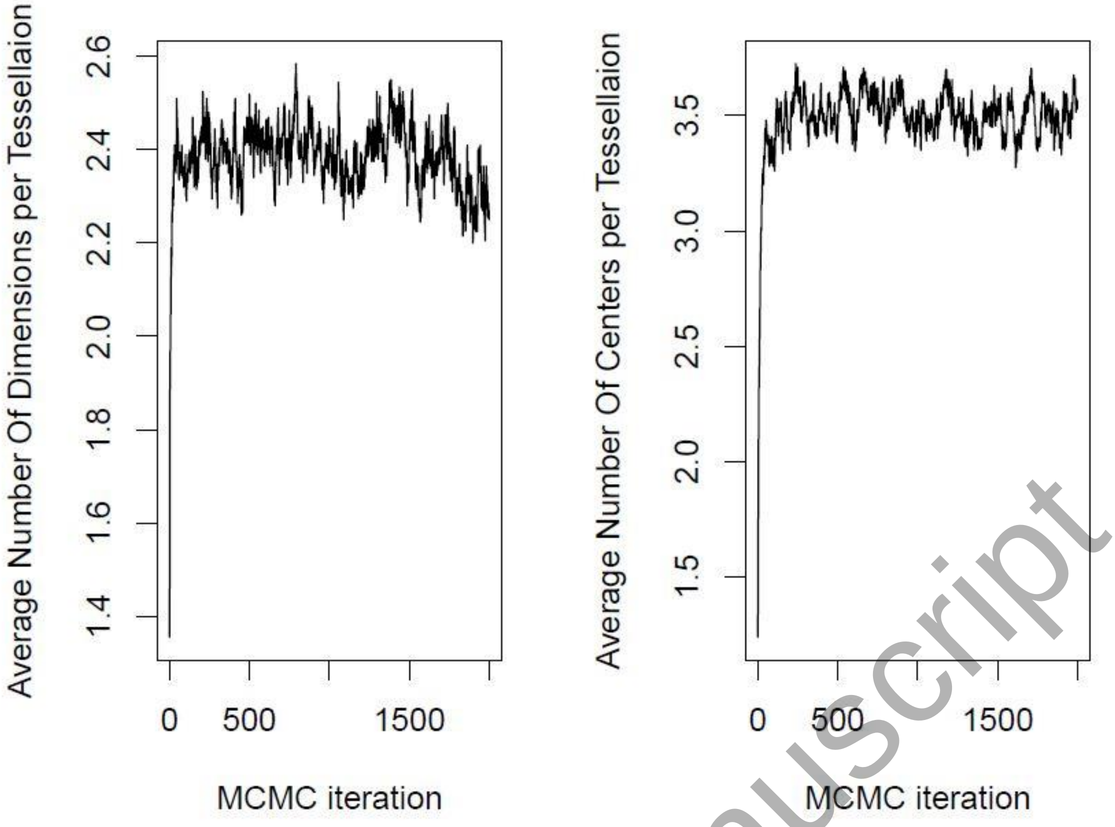

Figure 4: The average number of dimensions (left) and centers (right) per tessellation for each iteration.

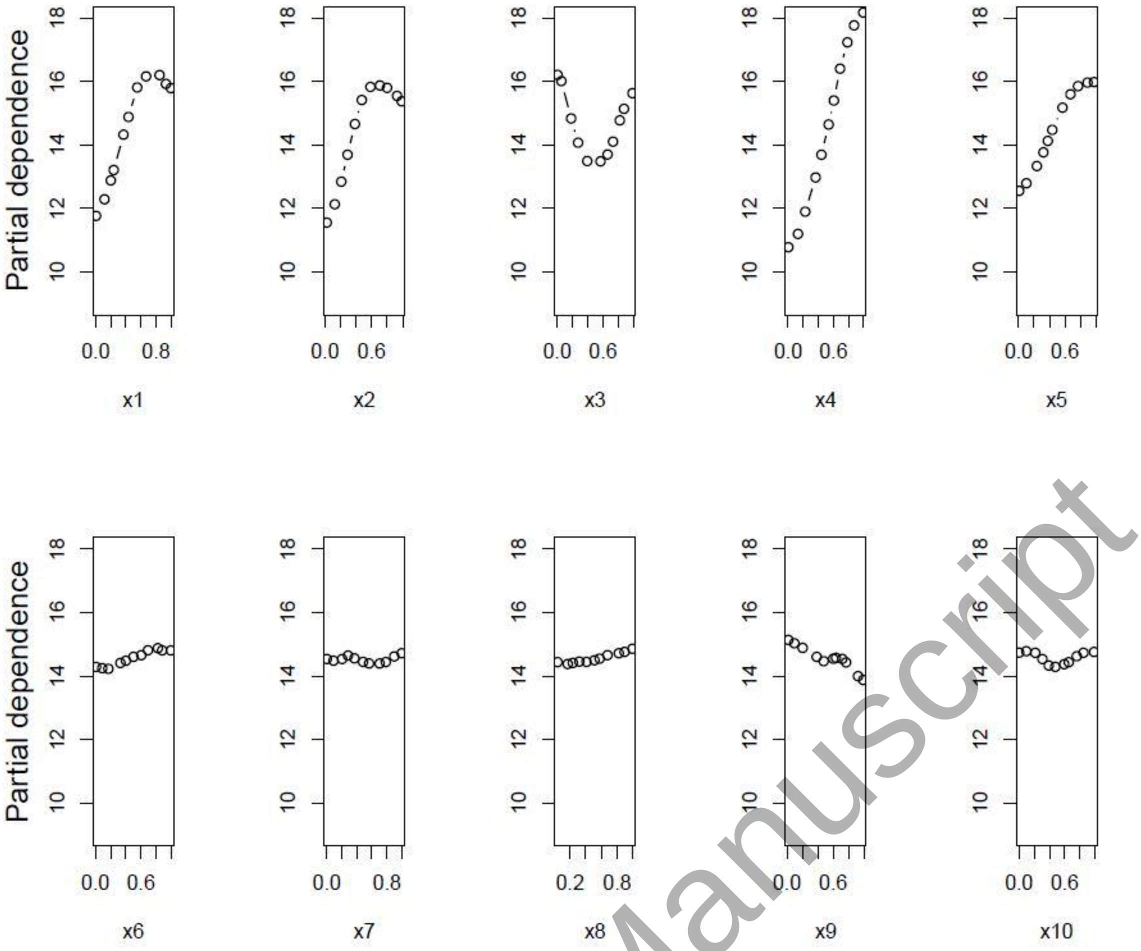

Figure 5: Partial dependence plots for the 10 predictors in Friedman function

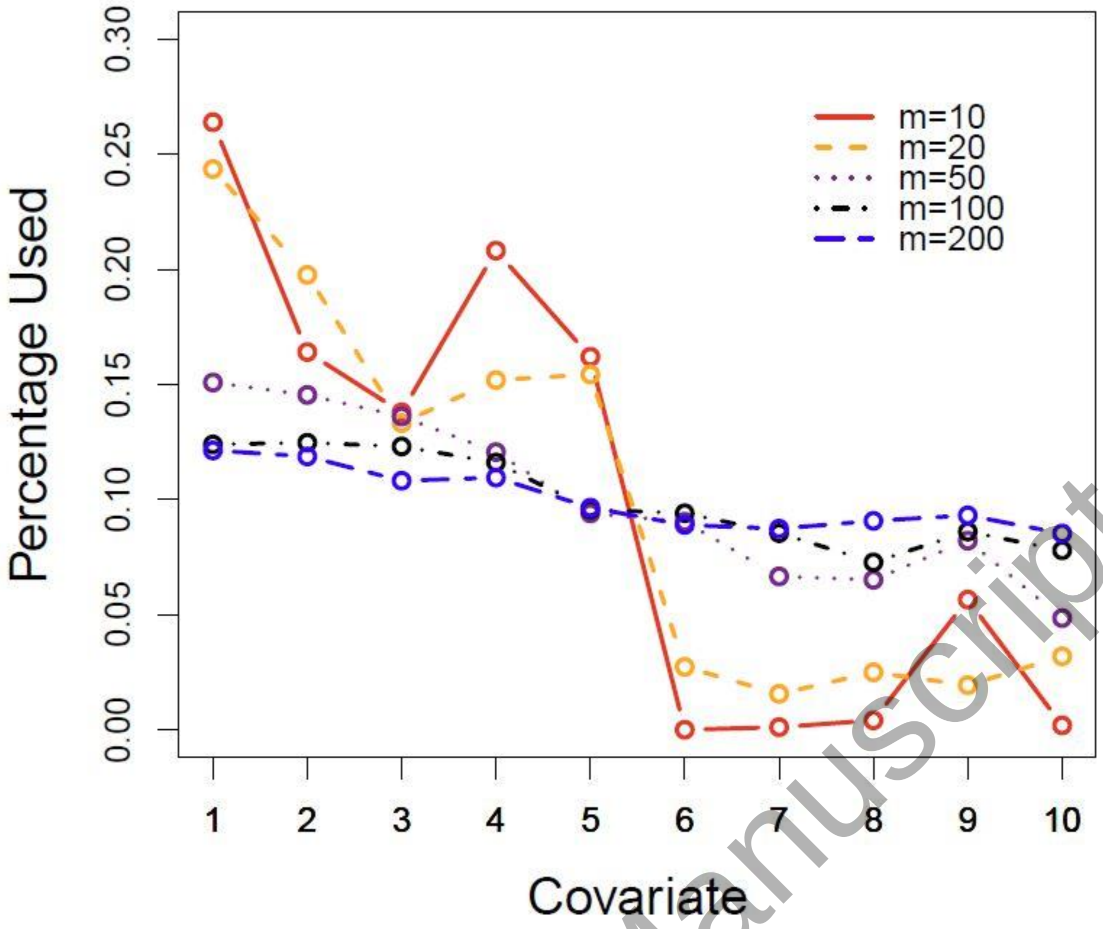

Figure 6: A graph showing the percentage of the covariates', $x_1,\ldots x_{10}$, that are most frequently included in accepted tessellations.

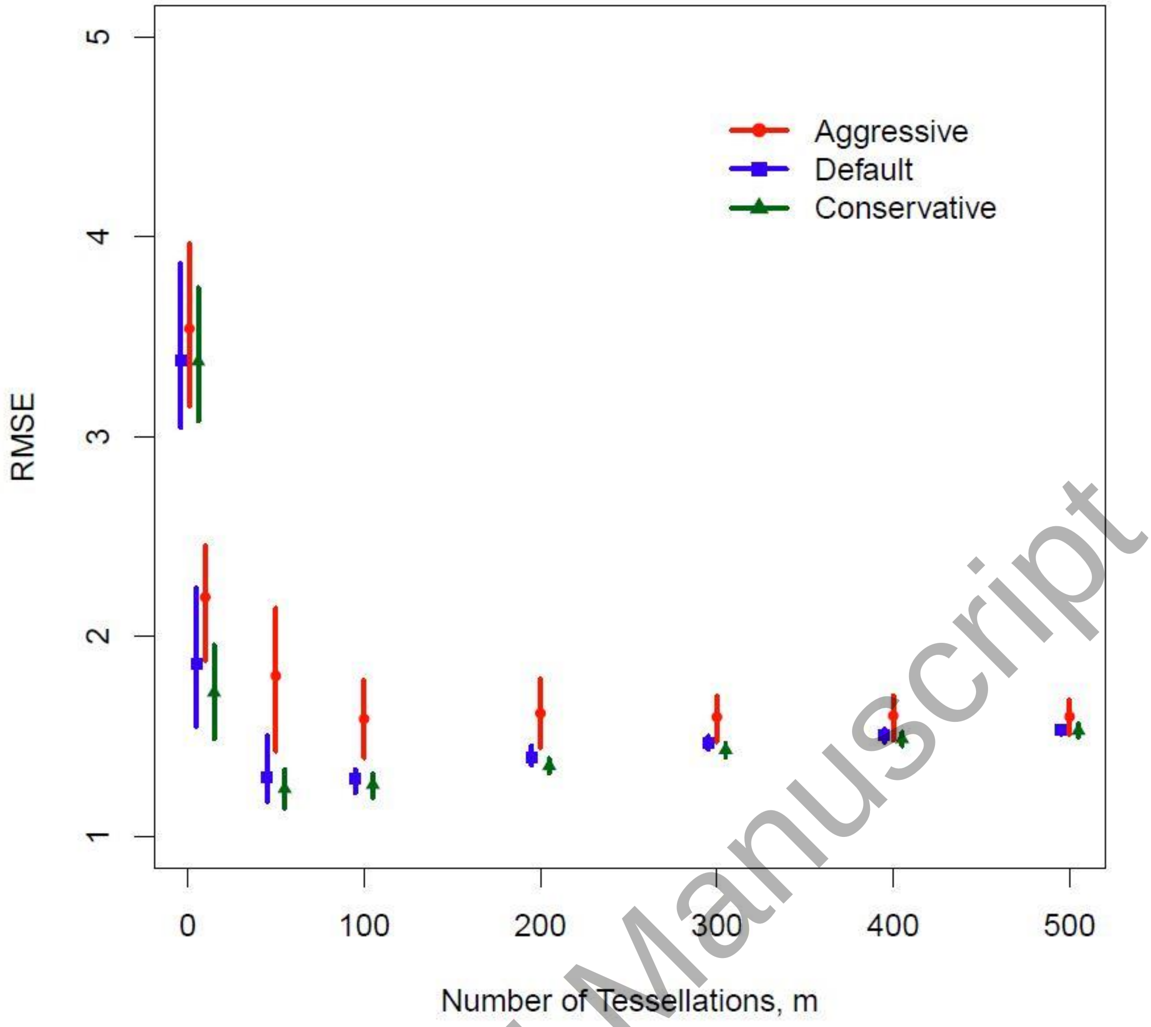

Figure 7: The RMSE value and 90% intervals as *m* is increased from 1 to 500 for three hyperparameter settings. An aggressive setting (circle) $(\nu, q, k, \sigma_c, \omega, \lambda_c) = (10, 0.75, 6, 0.8, 1, 5)$, the default setting (square) and a conservative setting (triangle) $(\nu, q, k, \sigma_c, \omega, \lambda_c) = (6, 0.95, 1, 1.5, 3, 25)$ have been considered. Note that that the points have been shifted minimally left and right to make the graph more interpretable.

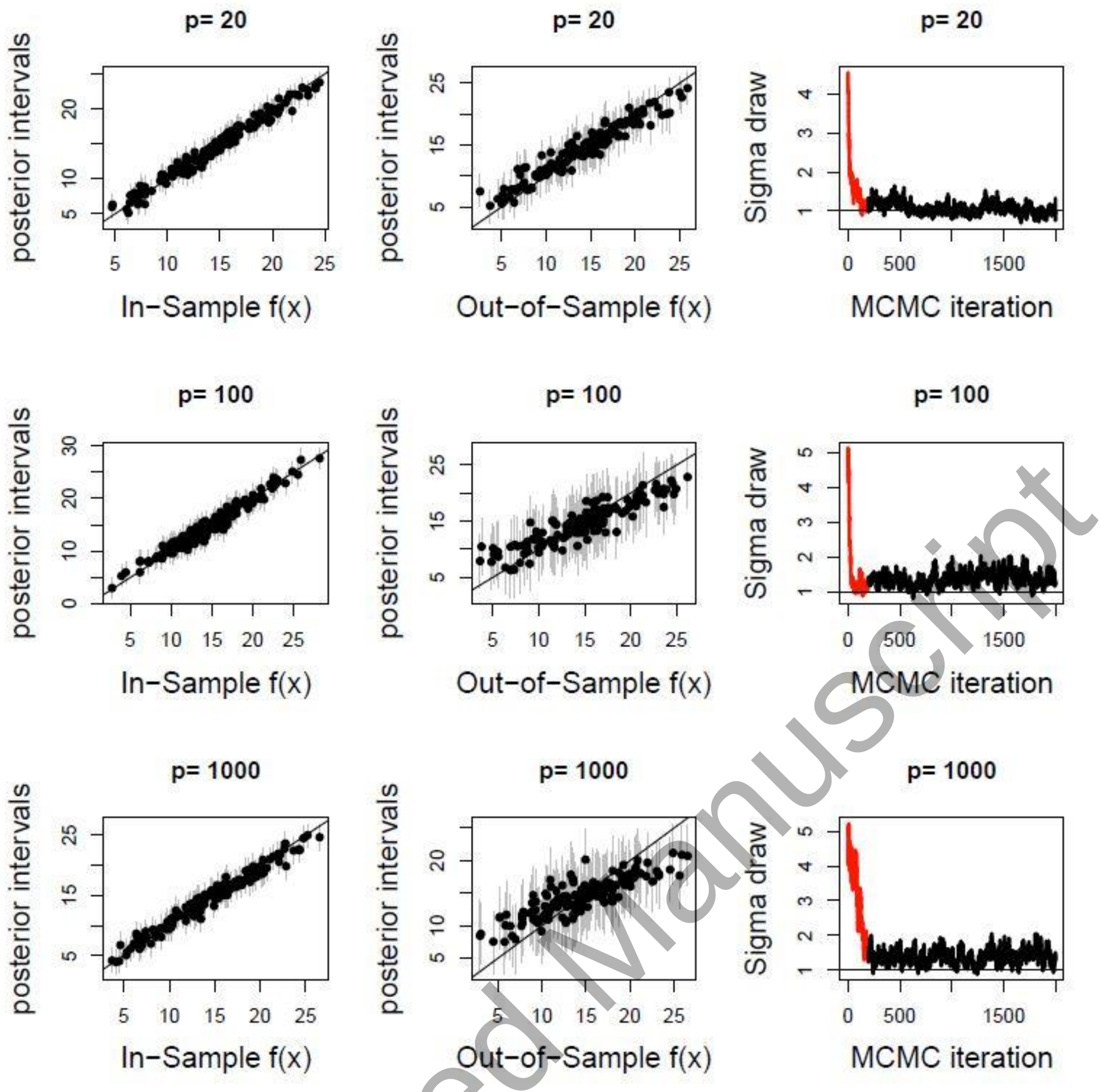

Figure 8: Inference about Friedman’s function for p = 20, 100, 1000 dimensions.

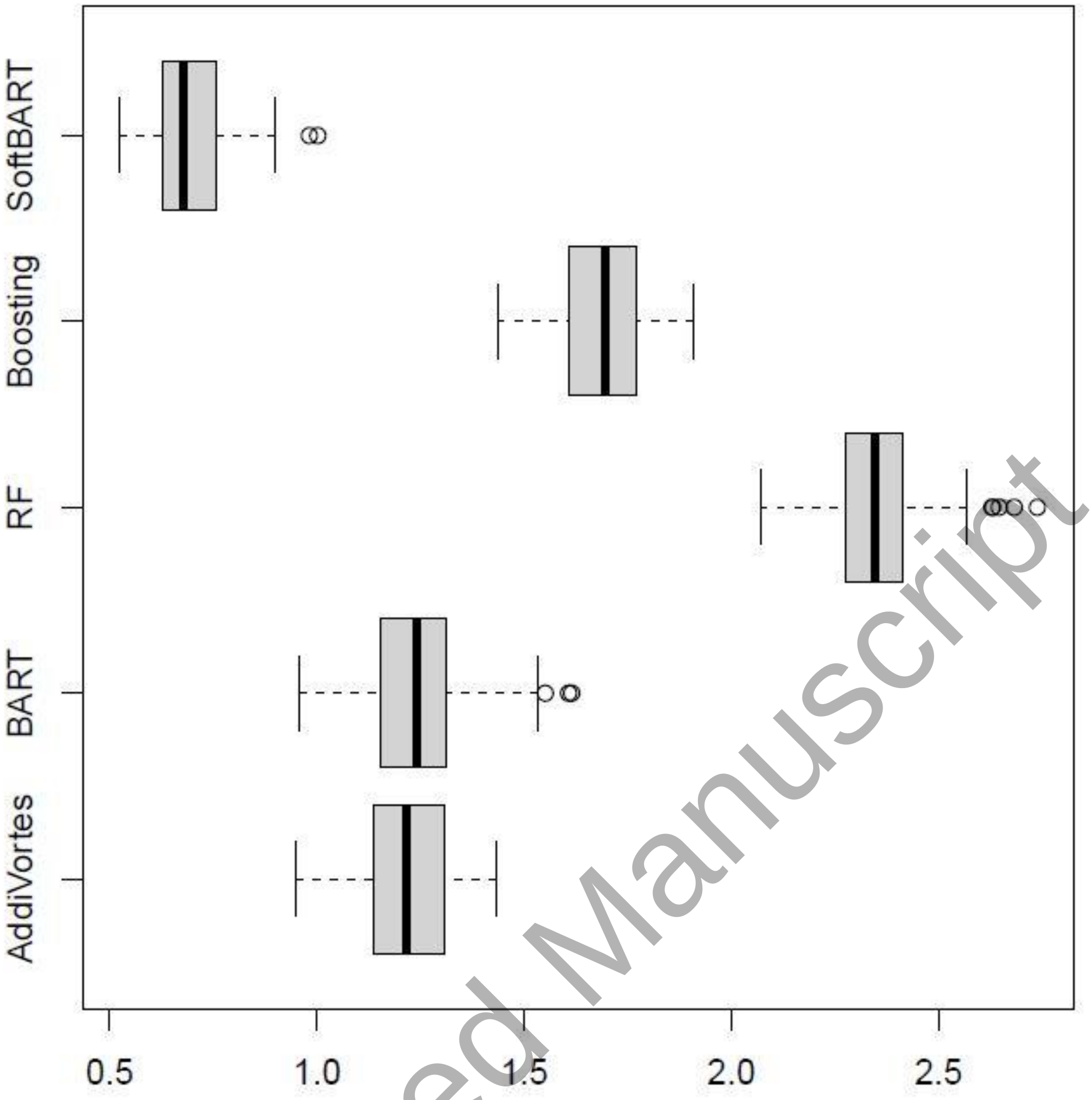

Figure 9: RMSE values for Friedman’s function for p = 10 covariates.

Table 1: A table showing the data sets used in our analysis.

Name	***n***	**Name**	***n***	**Name**	***n***	**Name**	***n***	**Name**	***n***
Abalone	4177	Baskball	96	Boston	506	Edu	1400	Enroll	258
Fat	252	Hatco	100	Labor	2953	Medicare	4406	Mpg	392
Ozone	330	Price	159	Rate	144				

Table 2: A table showing the cross-validation values for competing methods.

Method	**Parameters**	**Values considered**
Random forests	Number of trees	500
	% variables sampled to grow each node	10, 25, 50, 100
Gradient boosting	Number of trees	50, 100, 200
	Shrinkage (multiplier of each tree added)	0.01, 0.05, 0.10, 0.25
	Max depth permitted for each tree	1, 2, 3, 4
BART	Sigma prior: (ν, q) combinations	(3, 0.90), (3, 0.99), (10, 0.75)
	Number trees, m	50, 200
	μ prior: *k* value for σ_μ	1, 2, 3, 5
SoftBART	Number trees, m	50, 200
	μ prior: *k* value for σ_μ	1, 2, 3, 5
AddiVortes	# Tessellations: m	50, 200
	Sigma prior: (ν, q)	(6,0.85)
	μ prior: *k* value for σ_μ	1, 3
	Standard deviation of center location: σ_c	0.255, 0.8
	Probability weight for # covariates: ω	3
	Poisson rate for # centers: λ_c	5, 25

Table 3: (50%, 75%) quantiles of relative RMSE values for each method.

Method	$(\boldsymbol{50}\%, \boldsymbol{75}\%)$
AddiVortes-CV	(1.007635, 1.026602)
AddiVortes-Def	(1.011348, 1.028736)
SoftBART	(1.007709, 1.026700)
BART	(1.013905, 1.056225)
Random Forests	(1.036218, 1.100439)
Gradient Boosting	(1.068354, 1.184328)

Table 4: Coverage for the posterior intervals with increasing number of inactive covariates.

p	**In-Sample**	**Out-of-Sample**
20	95%	92%
100	97%	98%
1000	93%	90%