Tree Based Models Applied to Flying Fox Areal Data

Ben DeVries

Department of Mathematical Sciences
Montana State University

May 9, 2025

A writing project submitted in partial fulfillment
of the requirements for the degree

Master of Science in Statistics

APPROVAL

of a writing project submitted by

Ben DeVries

This writing project has been read by the writing project advisor and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the Statistics Faculty.

_____   _____
Date   Andy Hoegh
                             Writing Project Advisor

_____   _____
Date   Ian Laga
                             Writing Projects Coordinator

Hendra virus is a rare and potentially fatal disease endemic to Australia. Serological evidence of Hendra virus has been detected in Papua New Guinea as well. Flying foxes act as a reservoir host for the Hendra virus, which has been transferred directly to horses and indirectly to humans. Previous research suggests Hendra virus spillovers are driven by habitat loss and climate, which can force flying foxes out of there typical foraging patterns. Factors that influence flying fox feeding locations are thus directly related to public health, and species protection. We examine the impacts of land cover on flying fox feeding locations via tree based models; using radar data collected near Ballina Byron Gateway Airport in New South Wales Australia at the end of February 2024. Land cover data is from Google Earth Engine. A background discussion of several Regression Tree based algorithms is included, along with steps to prepare data for modelling. The results of our analysis of foraging sites provide some evidence of associations with man built structures, flooded vegetation, shrubbery, trees, and water. While land cover alone does not fully explain the feeding choices of Ballina's flying fox population, there appears to be some association.

Table of contents

1 Introduction

Flying foxes are a fruit bat, with around 65 species native to parts of Asia, Australia, and Africa. In this project, we examine a population of grey headed, and black flying foxes surrounding Ballina, New South Wales Australia. Our interest in flying foxes stems from the Hendra virus, which is endemic to the four species of flying foxes in Australia. This rare virus is zoonotic, meaning it can be transferred from animals to humans. There is a fairly low chance of bats transmitting Hendra virus directly to people. Humans who have been infected with Hendra virus have come into contact with horses exposed to waste/bodily fluids from flying foxes.

Spreading a disease through an intermediate host of another species is referred to as a disease spillover. In *Pathogen Spillover Driven By Rapid Changes in Bat Ecology*, Eby et al. [2022] identified loss of flying fox habitat and climate as influential factors in Hendra virus spillovers. Loss of habitat and climate both effect the availability of food and water, leading bats to venture into farmland and contaminate water/food supplies intended for horses. Understanding what factors influence flying fox feeding locations is thus naturally important to public health. For our analysis, we examine the feeding locations of the previously mentioned flying fox population, assessing whether any associations with land cover may be drawn.

Our data on flying fox feeding locations was accessed through a PostgreSQL database. Data was collected using a Robin radar, perched upon a hill within Ballina. The radar is able to track many objects at once, providing us with the flight

path, velocity, classification, and other details. Traditionally, human observers have been used to record location information for bats. Manual counting is obviously labor intensive, and potentially less precise than radar; especially after considering both black and grey-headed flying foxes are nocturnal. It's easy to imagine potential challenges in counting clouds of bats at dark. Additionally, the Robin radar observes a larger region than manual observation. Instead of counts at discrete observation sites, the Robin radar provides points over a continuous space.

When exploring associations, background information is always important. Baranowski and Bharti [2023] examined the potential spatio-temporal relationship between changes in land cover and flying fox roost sites in Queensland, Australia. In Kolkata, India, Roy et al. [2024] analyzed types of trees and urban structures that attracted flying foxes with ANOVA. Both of these studies utilized highly detailed land cover data that involved manual survey. They also focused on differences in similar types of land cover, whereas our land cover classes are broader but less specific. Our end goal is not to perfectly describe the effects of land cover, but rather to see what types of land cover may be associated with flying fox feeding locations. Focusing on exploration, we turn to tree based models due to their inferential capabilities and modelling flexibility.

We begin with a brief review of supervised learning before discussing Classification and Regression Trees (CART) [Breiman et al., 1984] along with several models inspired by CART. Next, our attention shifts to tools for obtaining and preparing data for our analysis. In this project we use Google Earth Engine to

obtain our covariates, and access the response from an archived PostgreSQL database. Finally, we apply Bayesian Additive Regression Trees (BART) [Chipman et al., 2010] to analyze the feeding locations of flying foxes.

2 Background

2.1 Supervised Learning

In traditional regression models we assume a specific relationship between the response, variables of interest, and controlling factors. Exploratory data analysis, background information, and model experimentation can bring light to potential relationships, but many problems involve unobservable dynamics that are difficult or impossible to capture via a parametric model. In some cases it is often necessary to take a more algorithmic approach to modelling. Unfortunately, many models in the algorithmic framework are referred to as black boxes, indicating the inner workings of a fitted model are difficult or impossible to explain in a meaningful way.

Whether we choose to focus on predictive or explanatory modelling comes down to our end goal, and often there is some degree of balancing the two. For example, let's think about a model to classify whether someone has a disease. Obviously we care about the predictive capabilities. If the model is less accurate than an informed guess, it's not particularly useful. Focusing solely on prediction and ignoring explainability, we might implement a support vector machine. The support vector

machine model would project patients measurements into a higher dimensional space and construct a hyper plane that separates the features into two groups, diseased/not diseased. Now if a doctor were to tell his patient, "We projected your measurements into a high dimensional space and unfortunately you fell on the wrong side of the hyperplane", the patient is almost certainly walking out.

Our doctor can soften the delivery; for example, they might adjust a patient's data and use the trained support vector machine to see how changes affect the predicted outcome. But patients are likely to ask how these changes impact their disease status. Logistic regression is a great example of an explainable model. Using the logistic regression model, the doctor might tell his patient "Looking at individuals similar to yourself, we've found an extra 30 minutes of exercise is associated with decreasing the odds of disease by a factor of X." Our doctor still has to interpret a mathematical formula and odds may be somewhat abstract to the patient, but there is a concise explanation.

Decision tree ensembles sit within the black box framework, but the explainability of individual trees has lead to establishing variable importance and interaction measures along with other useful tools for inference. It is still very difficult to interpret the effects of individual predictors in these models, as the hierarchical structure can make marginal effects very misleading. Still, variable importance measures combined with predictive accuracy make ensembles of trees a great tool for exploratory analysis.

Working with ensembles of trees, the models are not explainable in a

meaningful way as a individual prediction may involve hundreds or thousands of decisions. We forego explaining an ensemble's reasoning and instead ask two questions: How accurate is the model? And which variables were useful in the model? If our model can make good predictions, then variables deemed important have some sort of association with our response, even if it's indirect.

Prediction based modelling has led to the development of a variety of algorithms that focus on fitting the data as opposed to explaining it. These algorithms can be much more complex than linear models. *The Elements of Statistical Learning* [Hastie et al., 2009] was a key source for understanding algorithms and concepts in statistical learning. This text demonstrates situation where classic statistical models fall short, showing how complex models can solve intricate problems. Algorithms are explained in rigorous detail, while still providing intuition to implementations and capabilities.

Our background discussion begins with some notation and concepts in supervised learning. We then discuss the algorithms for several models, using simulation to demonstrate their use. In supervised learning problems, the goal is to map a set of features to a target. When discussing these components, we'll follow the notation used in [Hastie et al., 2009]. X denotes a features while Y and G denote the target for quantitative and categorical responses respectively. If there are N observations of p inputs, then $\mathbf{X}$ is an $N \times p$ matrix. The features for the ith observation are $\mathbf{x}_i$ while $\mathbf{x}_j$ denotes the vector of all observations for the j'th feature.

A common approach in supervised learning is to assume our features and target are observation of random variables with joint distribution $\Pr(Y, X)$. G would replace Y in a classification problem and $\mathbf{X}$ would replace X for multiple covariates. Letting $\mathcal{X}$, $\mathcal{Y}$ denote the supports of X, Y, we want to find a function $f : \mathcal{X} \to \mathcal{Y}$ that predicts Y reasonably well. To do this, we define a loss function $L(Y, f(X))$ which penalizes the prediction error on a training set $\tau = (\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$.

Our choice of loss function determines the optimal solution to the problem. The expected prediction error (EPE) is defined as $\mathbb{E}[L(Y, f(X))]$. Our overall goal is to approximate the function f that minimizes the EPE. Using squared errors for loss (L_2), the EPE is minimized by choosing f such that $f(x) = \mathbb{E}[Y|X = x]$ (The Elements of Statistical Learning, pg. 18 [Hastie et al., 2009]).

Now we need an algorithm to approximate f using τ. Choosing an appropriate algorithm will be heavily influenced by the availability and distribution of data. With a quantitative target, we might be able to assume an additive error relationship between Y and $\mathbf{X}$.

The additive error model is simply $Y = f(\mathbf{X}) + \epsilon$ where $\mathbb{E}[\epsilon] = 0$ and $\mathbf{X} \perp \epsilon$. Here we are assuming the process that generated Y can be approximated using $\mathbf{X}$ such that the error due to ignored features and other factors has mean zero. For data that was generated under an additive error model, a natural approach to approximate f is to use a linear combination of basis functions. A set of functions $\mathcal{B}$ form a basis if for every function $f : \mathcal{X} \to \mathcal{Y}$, there exists a unique linear combination of elements of $\mathcal{B}$ that is equal to f. The additive model is defined as

$Y = \beta_0 + \sum_{j=1}^{p} f_j(\mathbf{x}_j) + \epsilon.$

One limitation of an additive model is we assume no interactions between features. There's no reason we can't add interactions or functions of multiple covariates, but identifying these interactions is difficult. We might be able to fit a model with select transformations of covariates and every possible interaction, but this would almost certainly result in overfitting. Assuming a generating model with any form of noise implies even the optimal f will not predict every observation perfectly. A model is overfit when trends are ignored to explain slight irregularities in a sample. Discerning irregularities is highly subjective. Any decision we make based on observed data may lead to overfitting. To compare models predictive capability, we use data that has been held out from training to obtain a sample of prediction errors. With this sample, we can use some metric (usually based on our loss function) for model validation.

There are a variety of methods to perform model validation, but the two most common are hold out and K-fold cross validation. In hold out, some portion of the data is held out for testing and the rest is used to fit the model. If the model contains hyper parameters, we could split the data into training, validation, and testing. Fitting models to the training set, we can try various values for hyper parameters and compare model performance for predictions on the validation set. After we determine the optimal hyper parameter values, we refit the model on the combined training and validation set before using the testing set to assess the final fit. In some cases, additional tuning may be required after combining training and

validation as additional training data may change the optimal values of hyperparameters. Using training/testing splits is generally favored for computationally intensive problems with observations to spare. The primary disadvantage to hold out validation is we can't use as much data for training.

In K-fold cross validation, we get to train and test using all available data by first splitting the data into k folds of size $\approx N/k$. One fold is held for testing and $k-1$ are used for training. This is then repeated until all folds have been tested. A metric can be then used to select the model before fitting it to the complete dataset. K-fold makes the most of the available data, but is more computationally expensive. Additionally, K-fold does not assess the final model fit as parameter estimates will change as the model is refit. For models with low variance, representativeness of the available data is really the only concern. If there is a lot of variation in estimated parameters across folds, our cross validation becomes more of an assessment of the theoretical as opposed to fit model. This can still useful for model selection and refinement.

2.2 Decision Trees

Decision trees are a type of hierarchical model where the the feature space is partitioned into hyperrectangular regions, and a model is fit for each region. For regression tasks, we typically use mean only models. In this case the decision tree model is a step function. Unlike our mean only models for linear regression, we do not make any distributional assumptions. Without a distribution, we can't quantify

the uncertainty in our predictions. Additionally, the binary structure of trees changes the interpretation of effects. Small changes in a continuous feature $x_{i,j}$ may not effect the associated prediction $\hat{y}_i$. The mathematical relationship defining a linear model is highly informative, but rather rigid. Trees greatly outperform linear models with more complex data. A hierarchical structure naturally handles multicollinearity, and can help us find interactions between covariates. Even though we don't believe $\mathbf{X}$ is related to $\mathbf{y}$ by a step function, we can gain insight to the true relationship by a step function approximation. An example of a tree model is shown in Figure 1. Every decision after the top root node is conditioned on the previous decisions, allowing us to display the hierarchy graphically.
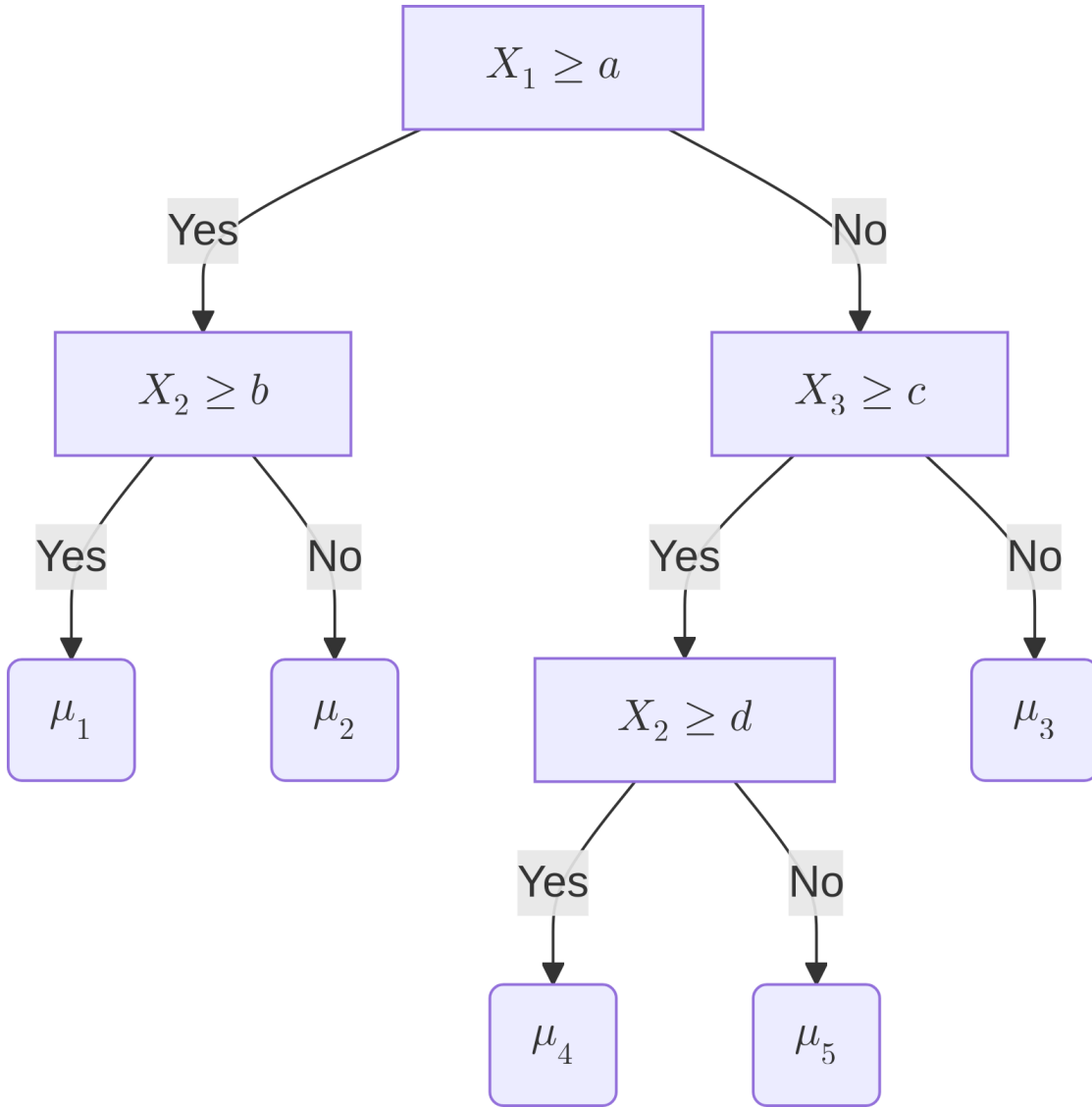
Figure 1: Visualization of binary decision tree

Step functions are very flexible. If there exists a function that can map the features of our training data to the observed response, then we can represent said function with a decision tree and achieve zero error over . With this flexibility, over fitting is a major concern. We can control the overall size and shape of trees with three hyper-parameters. First is the complexity parameter α that scales the penalty

which is based on the number of terminal nodes (mean only models). Next we have the minimum split and minimum bucket sizes. The minimum bucket is the fewest number of observations allowed in a terminal node. Similarly, the minimum split is the least number of observations allowed in one outcome of a binary decision. We generally choose the complexity parameter by growing the largest possible tree, then removing terminal nodes with fewer observations. Next we compare the trees via cross validation and choose the tree with minimal cross validated error. This process of choosing an appropriate tree size is commonly referred to as pruning. While pruning helps reduce the risk of over fitting, large regression trees are inherently unstable.

When we say trees are unstable, we are essentially saying that a small change to the data can result in a wildly different fit. We demonstrate the variability in the structure of trees with a simulation study. Data for the simulation study is generated under the model defined by equations 1-4.

$$y_i = \cos(108t_i \sin(108t_i + w_i)) \cdot \exp(\sin(108t_i \cos(108t_i + x_i))) + z_i \qquad (1)$$

$$W_i, X_i, Z_i \sim N(0, \sigma^2) \qquad (2)$$

$$t_i = -\frac{\pi}{18}, ..., \frac{\pi}{18} \qquad (3)$$

$$i = 1, 2, ..., 350 \qquad (4)$$

The rpart function developed by Therneau and Atkinson [2025] is used to fit our

CART model. A simulated testing data set displaying the true function along with the fit from a tree chosen by cross validation on training is displayed in Figure 2. Figure 3 displays the number of splits for the optimal complexity parameter chosen by cross validation.
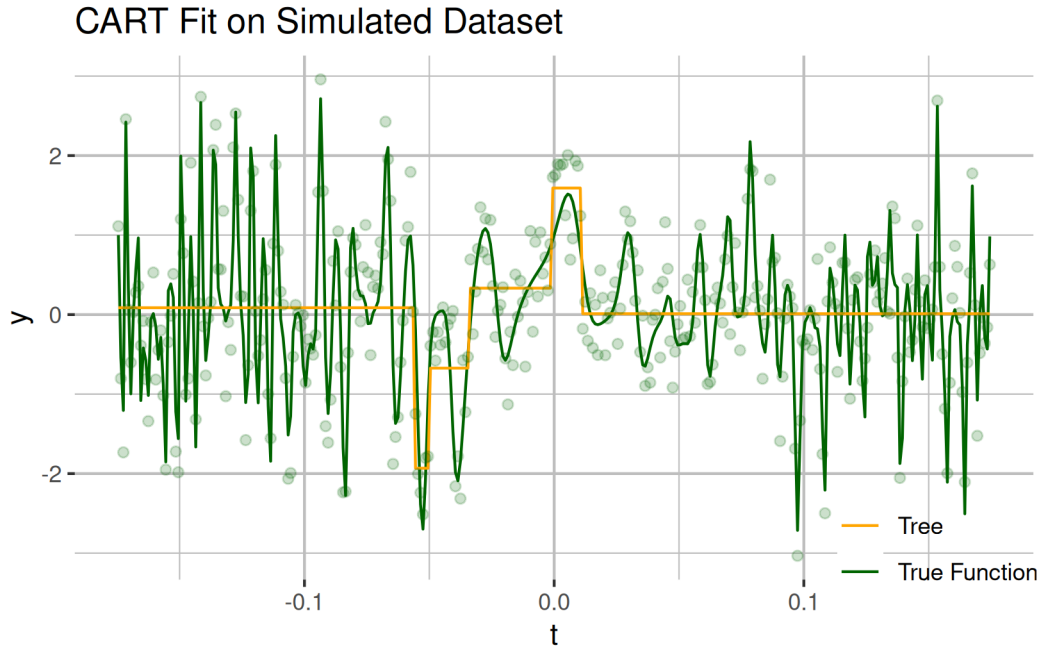
CART Fit on Simulated Dataset

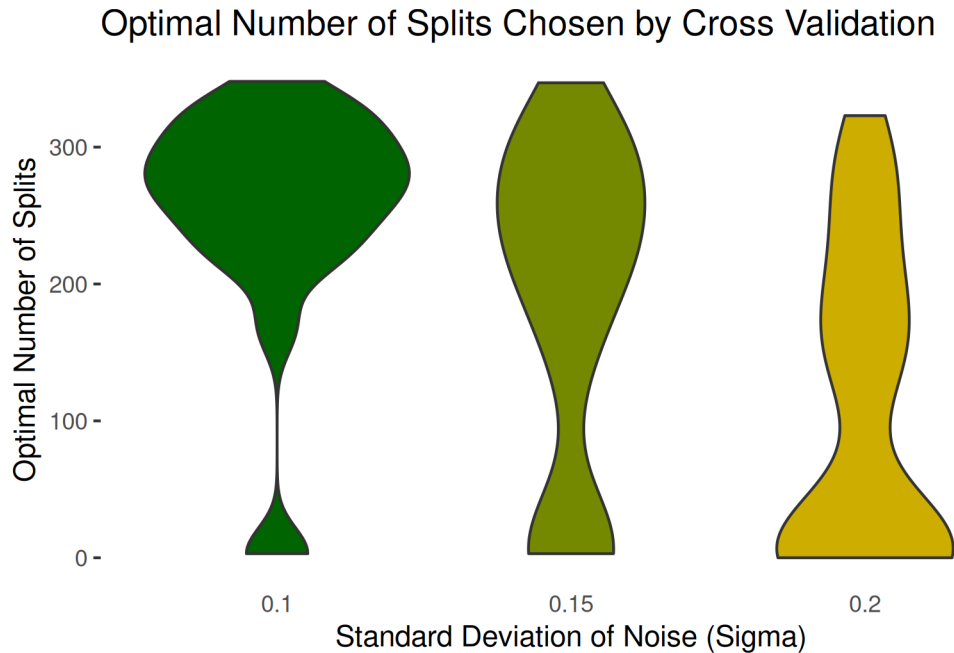Figure 2: Visualization of simulated data and predictions show a decision tree is simply a step function.

Figure 3: CART models can give similiar quality predictions with very different fits. Large variation in the structure of the model indicates inference can't be trusted with ugly data.

Decision trees are very unstable for this data. Across all three noise levels, the number of splits chosen by cross validation ranges from zero to over 300. The variability in the size/structure of trees suggest we can't have much faith in predictions from an individual tree. Slight changes to the data could result in a completely different tree structure, meaning our model may generalize poorly. A simpler model will be less variable, but more biased. To maintain low bias and reduce variance, we turn our attention to ensemble methods. Ensemble methods are quite prevalent in predictive modelling. By combining the predictions from several models, we can reduce the variance of predictions while capturing complex intricacies of the data.

The random forests algorithm, developed by Breiman [2001], is a CART-based

ensemble combining Bootstrap Aggregating (Bagging) [Breiman, 1996] and Ho's random decision forests [Ho, 1995]. Bagging is simply averaging the predictions of models fit to bootstrapped samples. When possible, we obtain $\hat{y}_i$ from averaging trees where $\mathbf{x}_i$ did not appear in training. These predictions are referred to as "out of bag predictions". Cross validation on out of bag predictions is not required as they are not based on the observed values of $\mathbf{x}_i$.

There's some intuition to the random forest algorithm. We know trees are flexible, and bagging will allow models to consider variations in the data. *The Elements of Statistical Learning* present several proofs demonstrating results that provide some intuition to the power of bootstrapping. On pages 271-272, Hastie et al. [2009] show that the distribution of bootstrap means for infinite samples from a normal population is equivalent to to the Bayesian posterior distribution with a non-informative prior. They also demonstrate that for the multinomial likelihood, the bootstrap distribution with infinite draws from the population approximates the Bayesian posterior assuming all outcomes are equi probable apriori. Later on pages 282-288, Hastie et al. [2009] show that the MSE of bagged estimates where samples are drawn from the population is less than or equal to the MSE of an individual model. For classification problems, Breiman [1996] shows that with infinite samples from the population, bagging can approximate the optimal classifier under certain conditions. This requires the majority of classifiers fit to bootstrapped samples to predict the correct class more often than not and relatively balanced classes. Otherwise, bagging can reduce accuracy. All of these results concerning infinite

samples are not very useful, but hint at the practical applications of bagging.

The main limitation in bagging alone is correlation between models. When averaging results from two trees T_1, T_2, the variance of our estimates will be $\frac{1}{4}(\mathbb{V}[T_1] + \mathbb{V}[T_2] + 2 \cdot \text{cov}(T_1, T_2))$. If only a few variables contain relevant information, we expect trees to make similar predictions. The random forests algorithm of Breiman [2001] combats the correlation between trees by taking the same approach implemented by Ho in random decision forests, [Ho, 1995]. Random decision forests average multiple decision trees fit to the same data. The key feature in the algorithm is the random subspace method where a random subset of the available predictors is drawn for each decision. Introducing randomness brings variety to the forest, resulting in weaker correlation between trees. The combination of random subspace and bagging makes Breiman's random forest a flexible model with low variance. The hyperparameters for the random forest model are the number of trees to grow, the minimum and maximum number of observations allowed in a terminal node, and the number of predictors to select for each decision. In Figure 5 we compare the MSE of models fit to data simulated in the same way as the previous simulation study. The rpart [Therneau and Atkinson, 2025] and randomForest [Liaw and Wiener, 2002] functions were used to fit models.
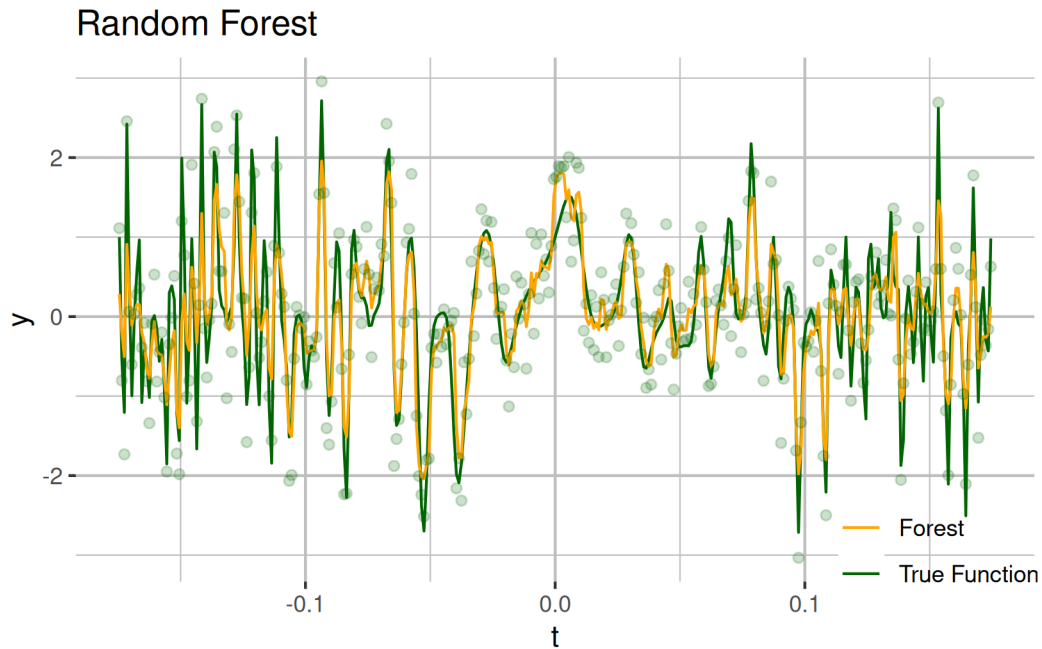
Figure 4: Combining many step functions defined by decision trees leads to smooth predictions with impressive accuracy and precision.
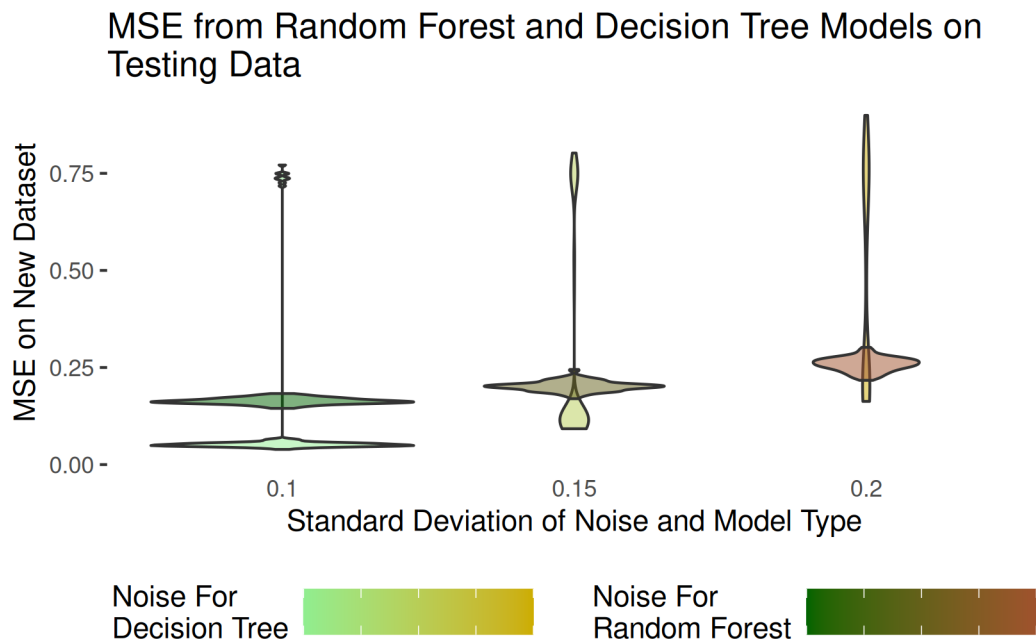

Figure 5: Even if individual trees give better predictions than random forests on this data set the majority of the time, using single trees for predictions is risky because of the large variance.

There is considerably less variation in the mean squared error of random forests fit to the same data as individual decision trees, but single tree models generally outperformed a random forest. Random forests are designed for data with several covariates. With a single predictor, the random forest algorithm degenerates to bagging with out of bag predictions. Bagging reduces the variance of predictions at the cost of bias, and thus the random forest predictions were often worse than predictions from individual trees. Pages 600, 601 of The Elements of Statistical Learning, [Hastie et al., 2009] display a proof showing that the bias of a random forest is the same as the bias of any individual tree in the ensemble. The randomization in the random forest algorithm implies trees within the ensemble are almost certainly smaller than a single tree fit to the data. In cases where we have access to several predictors, large trees will create complex hierarchies that are unlikely to exist in the true model. The following simulation study shows the random forest algorithm consistently outperform a single tree for data generated under a linear model with correlated covariates. Violin plots of MSE on testing data are displayed in Figure 6.
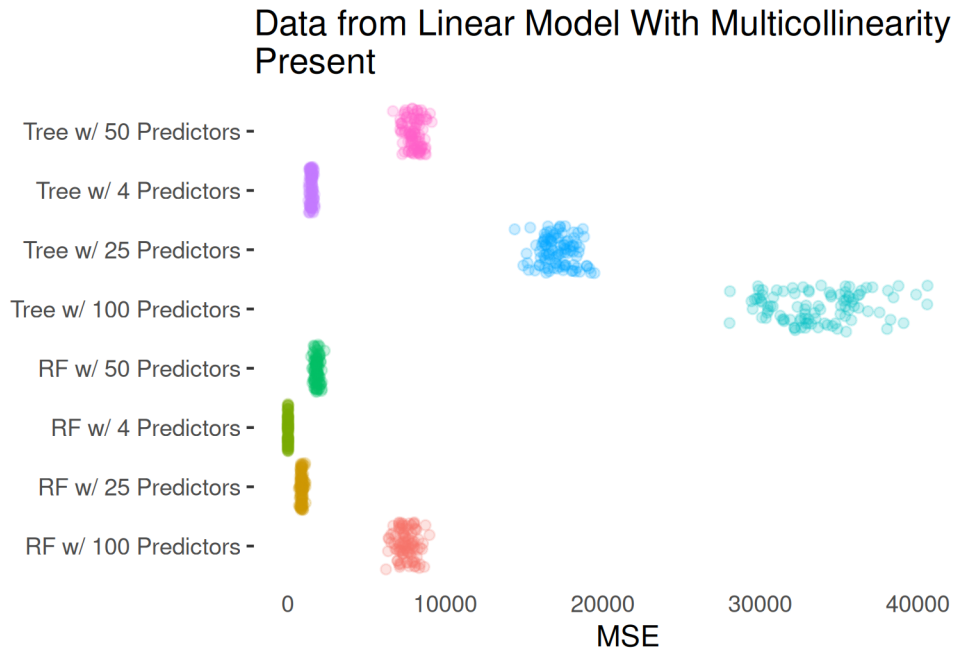
Figure 6: As relationships become more complex (i.e. more factors involved), mean squared error tends to increase. Generally, the increasing complexity has much less of an effect on random forests compared to individual trees.

2.3 Bayesian CART

Bayesian tree algorithms assume a stochastic process generates the tree, and a distribution for the observations within the terminal nodes. Randomness is used to explore possible structures that may increase the posterior. The flexibility of trees necessitates regularization of their structure, which can be easily achieved through the use of a prior. There are many ways to specify a Bayesian tree but, our discussion is limited to the Bayesian CART method proposed by Chipman et al. [1997]. In Bayesian CART, the search process begins with an initial stump tree, and proposes a new structure for the tree at random. For the proposal, there are four possibilities. A change step is proposed with probability 0.4. In the change step, a

non-terminal node is randomly selected and the splitting rule replaced. The new rule is chosen by randomly selecting a splitting variable X_j, and then randomly selecting an observed value of $\mathbf{x}_j$ as a cutoff. The same splitting rule cannot appear multiple times within the current tree. For the grow step of the search, a terminal node is randomly selected, a splitting rule is added, and a sibling for the chosen node is grown. The probability a grow step is 0.25 and the decision rule is chosen in the same way as in the change step. The third possible move in the search is randomly selecting intermediate nodes, and collapsing all nodes beneath. This step is referred to as pruning and is proposed with probability 0.25. The last possible move for a trees structure is node swapping. Parent and child intermediate nodes are selected and the rules are swapped. After updating the structure of the tree, new nodes are marked as splittable with probability $\frac{\alpha}{1+\beta\gamma^d}$ where $\alpha \in (0,1)$ and $\beta, \gamma \in (0,\infty)$ are hyperparameters and d is the depth of the selected node. Finally, terminal nodes are updated and the proposal is accepted or rejected. Chipman et al. [1997] describe methods for independent and correlated terminal nodes. With the proposal tree constructed, the Metropolis-Hastings algorithm is used to update the sequence of trees.

Bayesian CART may become stuck around a local mode where the next tree in the sequence provides little to no improvement. To combat this, a sequence of trees is generated until the change in posterior probability between trees in the sequence stabilizes. The algorithm is then restarted to search for another local mode. Convergence is tracked by comparing the posterior probability distribution of a tree

to it's predecessors, keeping track of which trees have the highest probability. The frequency of tree structures explored is ignored, making Bayesian CART a stochastic search as opposed to a Monte Carlo based method like in BART. Averaging trees that have converged to a local mode results in a model similar to random forests. Unlike random forests there is a distribution for the predictions.

2.4 Bayesian Additive Regression Trees (BART)

The other learning algorithm we explored was BART, or Bayesian Additive Regression Trees. For a continuous predictor, BART assumes $y_i \sim N(\sum_{b=1}^m g(\mathbf{x}_i; T_b, M_b), \sigma^2)$ where T_b is the b'th tree, M_b are the associated terminal nodes, and $g(\mathbf{x}_i; T_b, M_b)$ is the function that assigns $\mathbf{x}_i$ to μ_{lb}, $l \in \{1, 2, ..., |M_b|\}$. Unlike the random forest algorithm, the trees are not bagged and each tree has access to each feature. Another key difference is the final model is based on the sum of all trees as opposed to an average. BART models use the same stochastic process to define the tree structure as Bayesian CART, but differs in model specification. In Bayesian CART, trees that have high posterior probability are recorded, but the number of occurrences for each structure is ignored. This makes Bayesian CART a stochastic search as opposed to a full MCMC. In BART, draws from terminal nodes remain after the tree changes structure. We thus have a probability distribution over the structure of the trees. Placing a highly informative prior on the terminal status of nodes encourages smaller trees, resulting in faster convergence and less variable predictions.

The algorithm starts by growing M trees with a single terminal node (stumps). Gibbs sampling is used to iteratively update each tree conditioned on all other trees. Conditional updates are simplified by expressing $T_b|T_1, T_2, ...T_{b-1}, T_{b+1}, ...T_m$ as $T_b|R_b$ where R_b denotes the distribution of residuals from predictions excluding T_b. Summing trees grown conditionally on one another makes the contributions of individual nodes small. Chipman et al. [2010] state "... we can imagine the algorithm as analogous to sculpting a complex figure by adding and subtracting small dabs of clay." Conditional updates along with a highly informative prior on the terminal status of nodes allow the model to explore intricate relationships in the data, without overfitting.

BART is an adaptation of the original Bayesian CART algorithm, [Chipman et al., 1997]. With the smaller tree sizes, the pruning proposal is modified to only prune terminal nodes. Even with this modification, trees will still collapse to a stump throughout MCMC. Convergence is aided by the somewhat unrealistic assumption of independent priors for each tree and the standard deviation. The joint prior is given by

$$p((T_1, M_1), (T_2, M_2), ..., (T_m, M_m), \sigma^2) = p(\sigma^2) \prod_{b=1}^{m} p(M_b|T_b)p(T_b)$$

where $p(M_b|T_b) = \prod_{l=1}^{|M_b|} p(\mu_{lb}|T_b)$, and $p(\sigma^2) \sim \nu\lambda/\chi_\nu^2$ with hyperparameters ν, λ. The hyper parameters are selected by first choosing a point estimate $\hat{\sigma}$. By default, this is the residual standard deviation for a multiple linear regression model. ν is then fixed ($\nu \in [3, 10]$ recommended) and λ solved for by imposing the constraint

$\Pr(\sigma < \hat{\sigma}) = q$, where q is an additional hyper parameter. In the prior for a tree $p(T_b)$, the probability that a node at depth $d \in \mathbb{N}$ is non-terminal is given by $\alpha(1 + d)^{-\beta}$, $\alpha \in (0, 1)$, $\beta \in [0, \infty)$. A discrete uniform prior imposes the initial belief that each feature equally likely to be selected for a nodes splitting rule. Similarly, a discrete uniform across the observed values of the selected predictor serves as the prior for the cut point in the binary decision. A $N(|M|\mu_\mu, |M|\sigma_\mu^2)$ prior is assumed for each $\mu_{lb}|T_b$ where $|M|$ is the total number of terminal nodes across all trees. The hyperparameters μ_μ and σ_μ are chosen based on the data such that $\min(Y) = |M|\mu_\mu - k\sqrt{|M|}\sigma_\mu$ and $\max(Y) = |M|\mu_\mu - k\sqrt{|M|}\sigma_\mu$. In [Chipman et al., 2010], they recommend choosing $k \in [1, 3]$. The BART R package [Sparapani et al., 2021] rescales Y such that $Y \in [-0.5, 0.5]$ and chooses

$$\mu_\mu = 0 \implies \sigma_\mu = \frac{0.5}{k\sqrt{|M|}}.$$

Two differences between BART and traditional Bayesian models are the number of trees m is fixed and data is used to inform the priors. These features push BART to be more of a non-parametric method, but with many of the features of Bayesian models. Inference is still derived from the posterior and hyperparameters can be tuned using a testing set with the goal of bringing prediction intervals to the nominal level. An example of a BART model tuned for nominal coverage to simulated data is displayed in Figure 7. The poor fit can likely be explained by non-additive noise in the generating function. BART is incredibly, flexible but at the end of the day it is still an additive model.
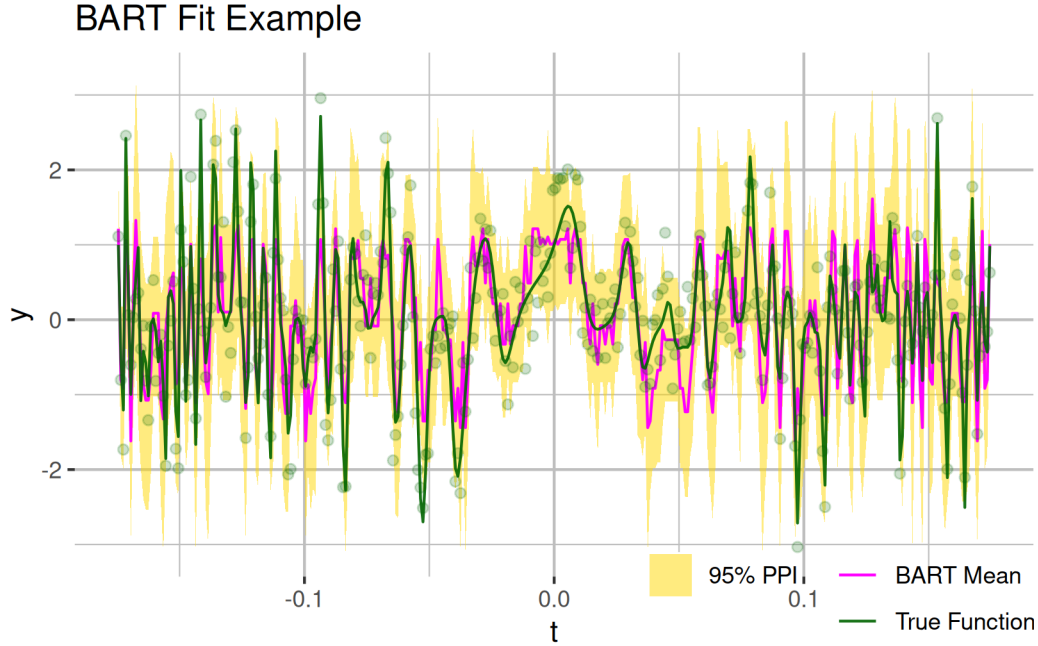
Figure 7: Visualising the predictions from BART, we see particularly peculiar predictions around $t = 0.05$. Note that the green points are from the training data, yet the mean appears well below. The non-additive noise in the generating model violates BART's assumption of normally distributed errors. Using a highly informative, misspecified prior on the standard deviation stretches the posterior predictive distribution to nominal coverage. As a result, the mean is pulled away from local observed values. The hyperparameter values used here are $\hat{\sigma} = 2SD(y)$, $\lambda = 220$, $k = 0.5$, $\alpha = 0.99, \beta = 0.5$, $q = 0.99$. A total of 250 trees were used.

Ensembles of trees are not used exclusively for prediction. Two inferential tools that are commonly used with both random forests and BART are variable importance and variable interaction measures. We briefly describe these measures, using [Inglis et al., 2024] bartMan R package. Notes on variable importance and variable interaction measures are based on the documentation. Similar measurements have been useful for inference from random forests, but the calculations are different.

Variable importance can be measured by looking at the proportion of decisions

in which a variable is used. For BART, the structure of the trees varies across iterations of MCMC, so we define $c_{r,k}$ as the number of times variable r is used as a splitting decisions within the k'th posterior sample. $c_k = \sum_{j=1}^{P} c_{r,k}$ is thus the total number of decisions for the k'th posterior sample. Variable importance is then measured as $\text{VImp}_r = \sum_{k=1}^{K} \frac{c_{r,k}}{c_k}$. Thinking of the extreme cases, with covariates that provide no information about the response, we should simply estimate f as the mean of the observed data. BART will make few decisions in this scenario as proposals will provide little to no improvement to the likelihood. If we had some variable that completely explained our response, BART may consistently make decisions off values of said feature.

Similar to how variable importance is measured, we can assess the potential for interactions by calculating the proportion of successive decisions concerning two variables across posterior samples. Individual decisions/sequences of decisions do not have much meaning on there own, making these algorithms hard to explain. It makes sense though that if a variable is consistently used, it must be able to explain the response to some degree. These tools can be used to interpret a BART model, or to help with variable selection for another model. Figure 8 displays a matrix of variable importance and interaction measures for data simulated under as

$Y = X_1(X_2+X_3+X_6)+X_2 \cdot X_6+\sin(X_1)+\cos(X_2)+\log(|X_3|)+X_4^2+\lfloor X_5 \rfloor+\lceil X_6 \rceil+\epsilon.$

We see all interactions are found, but the plot suggests some evidence of an interaction between X_3 and X_6. The coefficient of variation is a measure used to compare one variable to another and does not have much meaning on it's own. Like

any other measure in statistics, results must be scrutinized.
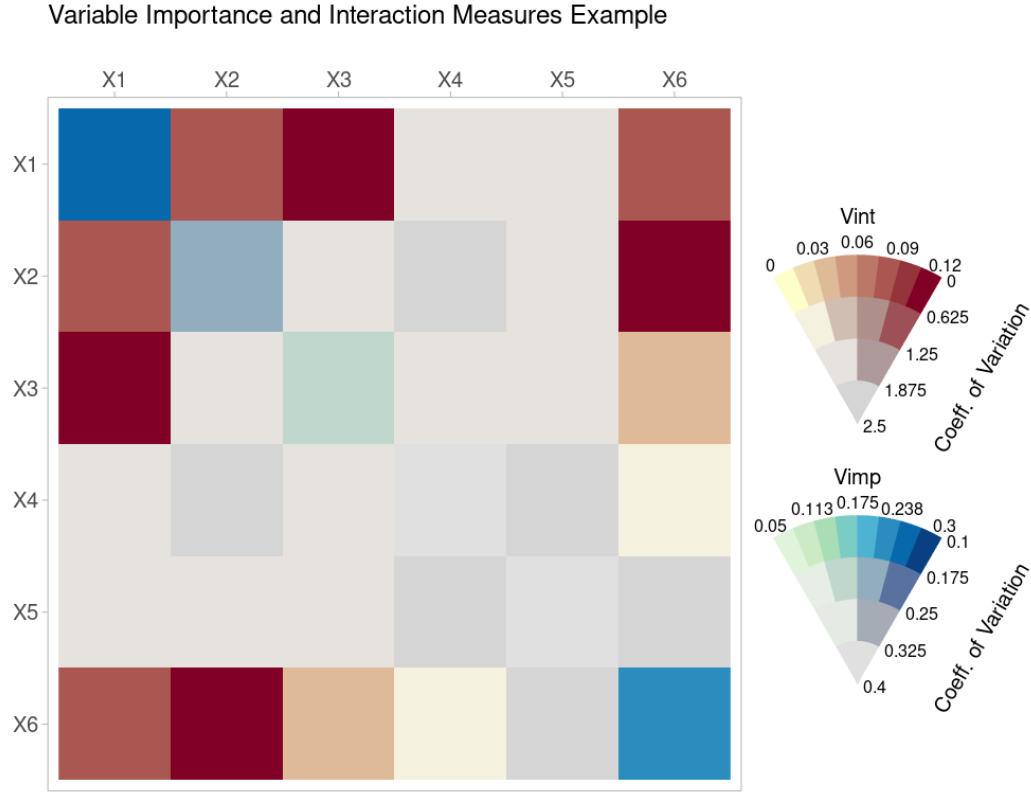

Figure 8: The matrix of coefficient of variation values for variable importance and interaction measures shows BART makes reasonable selection decisions in a contrived example. Variable importance and interaction measures are useful tools for inference and exploratory analysis even if they must be scrutinized.

Before we move on to the data, we briefly discuss diagnostics for BART. Unlike the random forest algorithm, a BART model for a continuous response assumes normally distributed errors and constant variance. When using a BART model, we need to assess the normality assumption, model fit, and convergence. The bartMan R package [Inglis et al., 2024] provides a convenient tool for diagnostics. Figure 9 shows the diagnostics for the same model used for Figure 8. The Q-Q plot and Histogram both indicate normality was reasonable as expected. We see constant

variance is reasonable as well in the Fitted vs Residuals Plot. The actual vs fitted plot shows BART fits the data quite well as the observed and fitted values are quite close. Variable Importance intervals shows BART correctly identified all variables as important as the intervals do not contain zero. The trace plot for σ shows the standard deviation has converged reasonably well. BART models have many parameters, and it is not unheard of to avoid assessing the convergence of each tree. The thought behind this is we don't care about any of the individual trees, so as long as our predictions have converged the model fits reasonably well. Changing the number of trees, along with α and β can help with convergence.
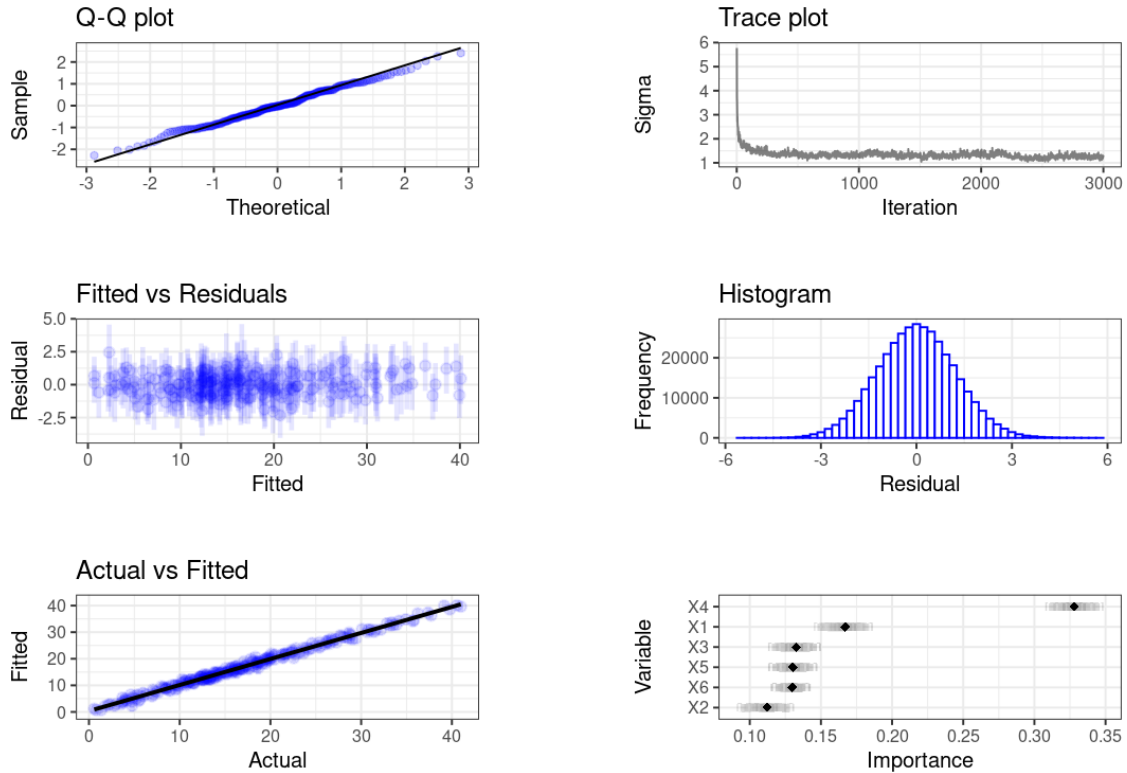

Figure 9: Diagnostic plots for BART show the model easily handles interactions and non-linear relationships. Like any other regression model, our inferences will be influenced by how well the model fits the data.

3 Data

Before we can apply any of the models discussed to assess the importance of land cover on flying fox feeding locations, we need to extract and transform the data. The flying fox locations themselves form a point pattern, which can't be modeled directly by any of the algorithms discussed. Point patterns are a realization of a point process, defined by a random number of observations in random locations. Modelling point pattern data requires estimating the mean occurrence rate across an entire surface. We're not particularly interested in the precise locations flying foxes feed so modelling a point pattern is overkill. Instead, we focus on the land cover characteristics of larger grid cells to examine the potential association with feeding locations. In this section we discuss the tools used to obtain and process our data.

3.1 Raster Data and Google Earth Engine

Rasters are used to store data with spatial components. Each pixel has an (x, y) coordinate, and a variable of interest. We can visualize rasters by mapping the variable to a color. Gradient scales can be used for continuous values. For this project, we use rasters pulled via Google Earth Engine (GEE), [Gorelick et al., 2017]. GEE has a large database of satellite images and rasters of estimated attributes. Many earth engine datasets contain images describing different attributes of the land. These attributes are labelled bands.

An important component of spatial rasters is the coordinate reference system

(CRS). We can think about the satellite images as pictures of a sphere. Using the position of the camera and other factors such as distance to the earth, (x, y, z) coordinates can be assigned to each pixel. For spatial data, we need to know how far apart locations are on the sphere. Looking at a picture of a sphere, we can't determine how far apart points are due to the curvature. Gauss's Theorema Egregium implies that no 2d map perfectly portrays the distance between points. We need to choose a CRS that preserves the distance between points for our area of interest reasonably well. The location on the earth and size of the area determine this choice. Each raster has a pre-specified CRS that may or may not be suitable, but rasters can always be projected into another CRS.

Working with data from earth engine we can either select an individual image, or an image collection. An image corresponds to a specific band and time, while the image collection is the entire data set. When either is selected, we have access to every time and location available. With an image, we can clip area of interest, masking data outside this region. With an image collection, we can filter the dataset to a specific region and time period. After selecting a band, an individual image can be chosen. Alternatively, we can produce an image based calculation over the filtered time period and region. Images can then be exported to Google Drive and loaded into R with the raster package, [Hijmans, 2025]. After loading the raster, we can create a long formatted data structure for modelling.

3.2 Land Cover and Dynamic World

One category of datasets on Google Earth Engine is land cover. Land cover describes natural and developed features of the earth as classes. A similar but distinct class of datasets is land use, which describes how humans use the land. There are various methods to estimate land cover, and many models estimate different classes. Additionally, different datasets span various regions and time periods. The majority of land cover datasets are annual and not quite up to date. Estimates are produced in a multi-phase modelling process, considering many images along with external prior information. Working with recent data from Australia, there are fewer options, especially after excluding low resolution datasets. In this project, we utilize data from DynamicWorld V1 by Brown et al. [2022]. Dynamic World is a daily high resolution land cover dataset derived from Sentinel-2 satellites. The estimated probability of each class in dynamic world is the output from a convectional neural nets predictions on a single satellite image.

Within Dynamic World there are ten bands, each containing a collection of images at a 10 m resolution. In our analysis, we use eight relevant bands that each contain the estimated probability that a 100 m^2 region of the earth corresponds to the chosen band. Unsurprisingly, daily high resolution data is rife with missingness. To combat this, we compute the median land class over a selected period of time. Missing values are ignored in the calculation by default. This also helps slightly with the fact that Dynamic World estimates are based on an individual day. Weather and atmospheric factors may influence estimates so looking at the closest day may

not be advisable. We don't expect major changes to land cover in a short period of time so this is reasonable. An individual image can then be produced based on some function of the images. A panel of the exported rasters is displayed in Figure 10.
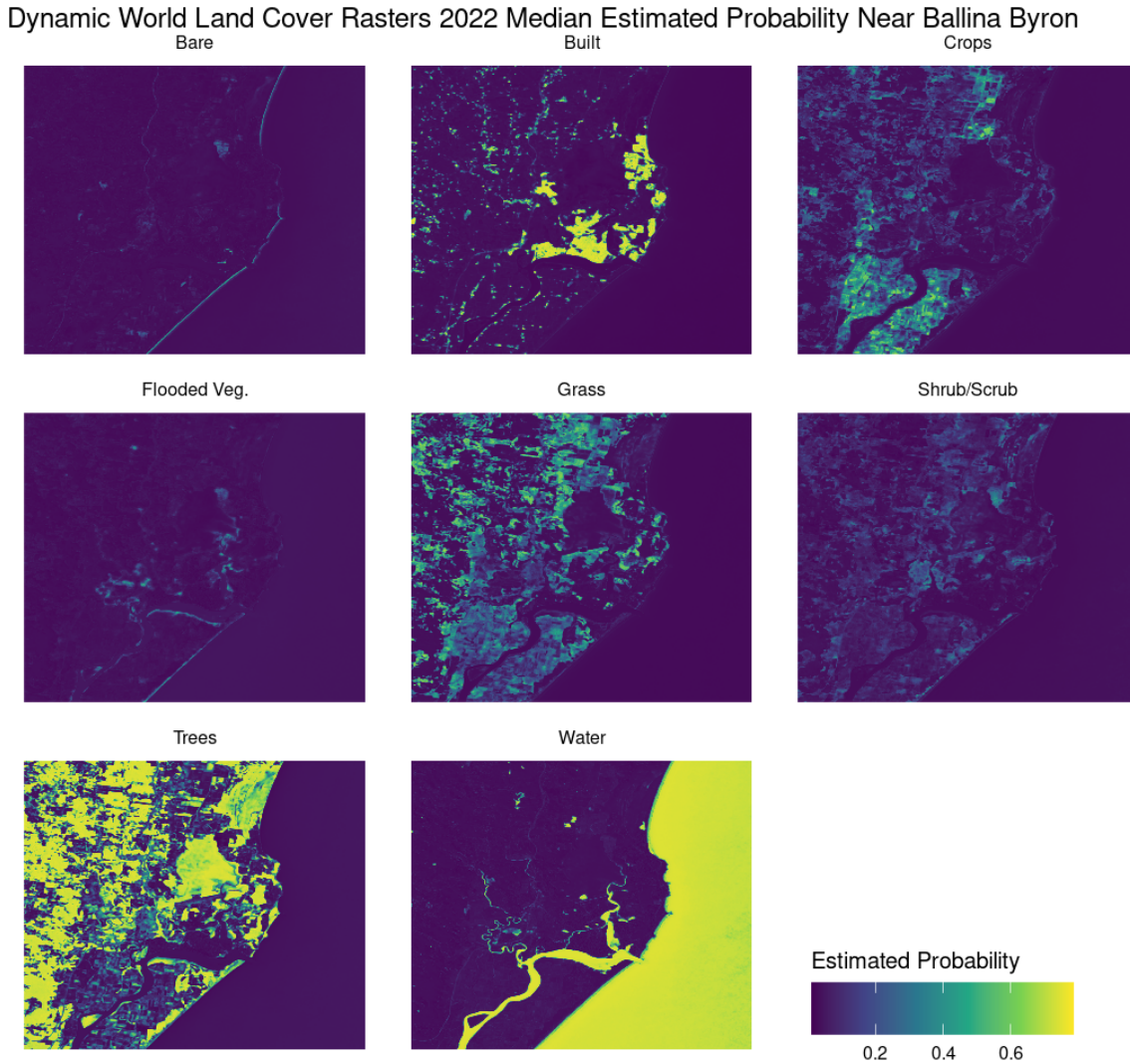

Figure 10: Dynamic World Exports from Google Earth Engine

3.3 Data Processing

The radar data used in our project was extracted from an archived PostgreSQL database with three days of observations. The radar itself is recording data day and

night, and sees many objects and organisms other than bats. Birds and bats are essentially the same to the radar. Our first step in processing the data was to filter observations by time and class. Using SQL, we construct a table of flight paths for the three sizes (classes) of bat where the last observed location is recorded between 12:00 am and 5:00 am. The flight paths are recorded as linestrings, a geometry representing the path through a sequence of locations.

Using our table of linestrings, we define an area of observation for the bats with the ST_ConcaveHull from PostGIS. The concave hull function takes a geometry or table of geometries, and shrink wraps a polygon around them. There is a parameter to control the convexity, which we set to 0.99, allowing minimal concavity. The polygon returned by ST_ConcaveHull along with the final observed location of bats were then exported for additional data processing.

Any form of spatial data can be transformed into areal data by defining a grid and aggregating observations within the cells. Before aggregating flying fox counts, we need to define a grid. Returning to Google Earth Engine, the previously defined boundary is imported, an enclosing rectangle defined. The coveringGrid function is then used to define our grid. The grid can be filtered based on our area of interest. Using the filtered grid we, sample pixel values from dynamic world to obtain the mean probability for each class over the grid. Resampling in earth engine is much easier than manually averaging as we don't have to worry about weighting averages when grid cells don't align with the cells of rasters. Without paying additional fees, the number of cells that can be resampled is capped at 5000. For a larger area, the

grid could be partitioned into blocks to resample in batches. At a scale of 289.6 m, our grid contains 4748 observed locations. There is slight variation in the size of cells, ranging from $73,212$ to $73,381$ m^2.

Our gridded area of interest was then imported into R along with the three nights of radar data. Using the lengths and st_intersect function from the sf package [Pebesma, 2018], we can easily obtain flying fox counts for each grid cell. Figure 11 shows our processed data rasterized.
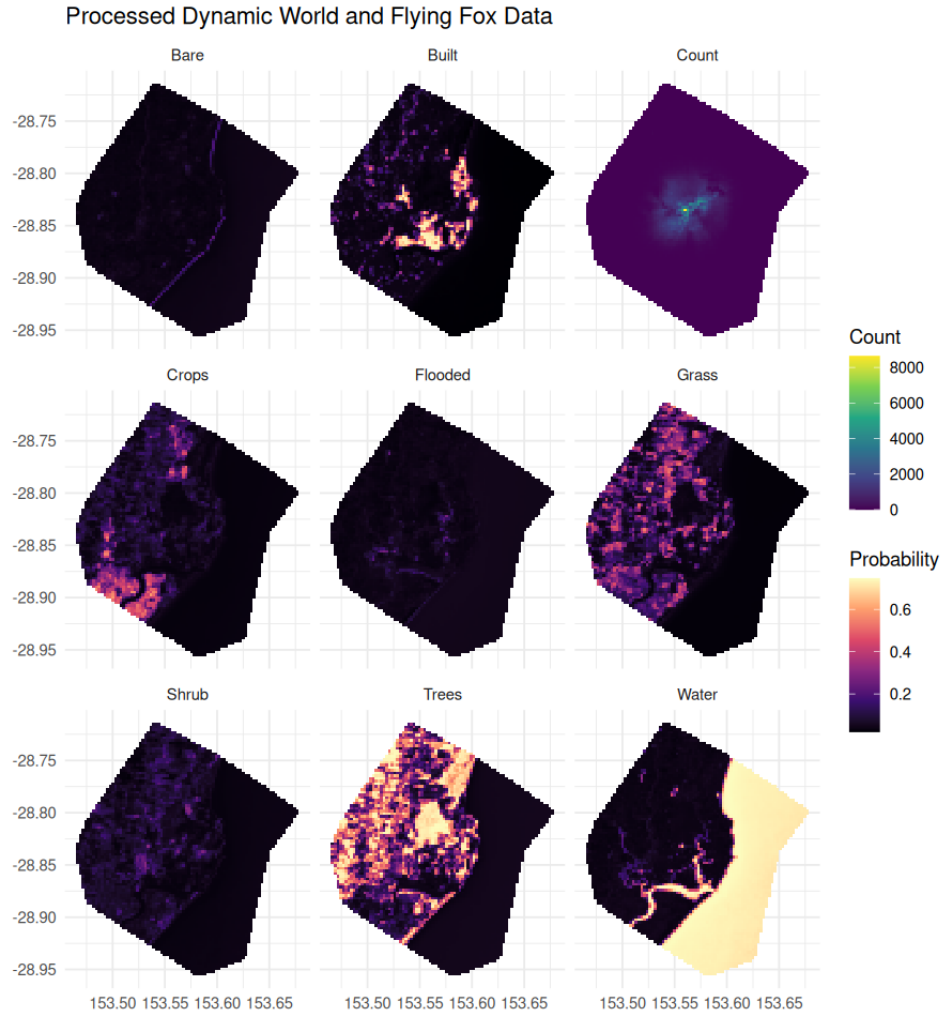
Figure 11: The correlation matrix indicates flying fox counts have weak to moderate correlation with land the various classes of land cover. It's still possible that our model will be able to make good predictions as interactions and non-linear behavior can make correlation missleading.

4 Modelling and Results

4.1 Exploratory Data Analysis

Our modelling process begins with a brief exploratory data analysis. While we are not specifying a parametric relationship between flying fox counts and land cover, exploratory data analysis may provide insights that help interpret results. A correlation matrix for the dynamic world data and flying fox counts was created with the ggcorrplot function, [Greenwood, 2025] and is displayed in Figure 12. Weak to moderate correlation is seen between flying fox counts and Built, trees, water, and shrub. Water shows a negative correlation with flying fox counts, which can be explained by flying foxes travelling to islands. It shouldn't be hard for BART to figure out that bats will not feed in the ocean as the estimated land cover probabilities other than water are quite low in these areas. While several of our covariates display fairly low correlations with flying fox counts, non-linear relationships and interactions may make features such as bare relevant.
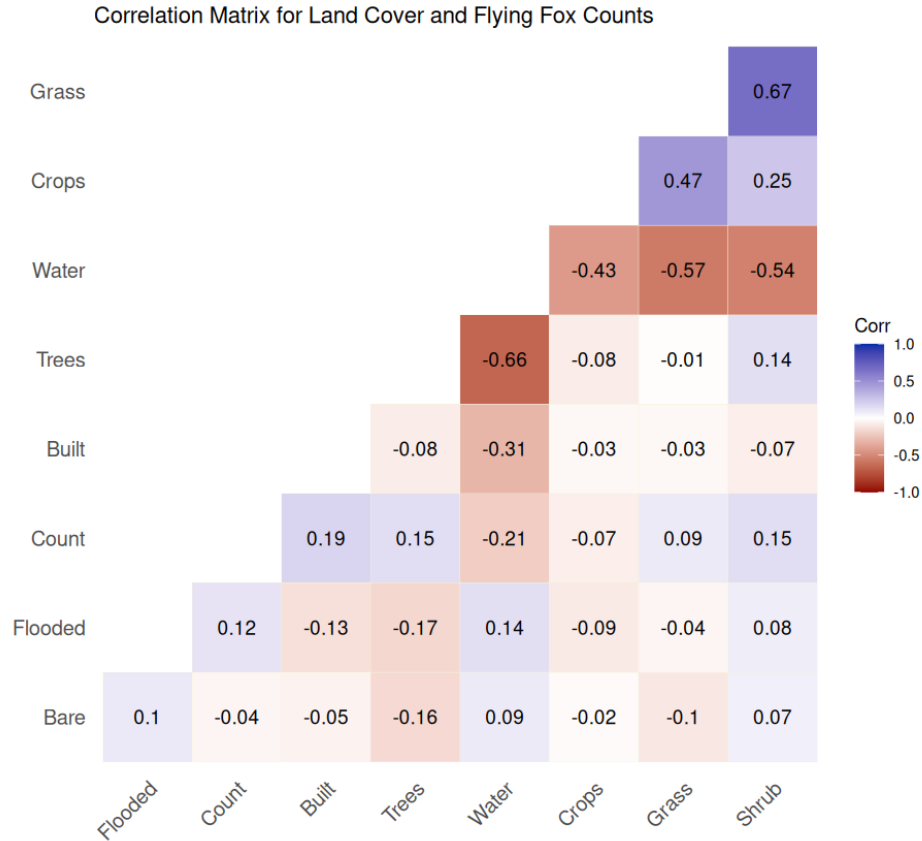
Figure 12: The correlation matrix indicates flying fox counts have weak to moderate correlation with land the various classes of land cover. It's still possible that our model will be able to make good predictions as interactions and non-linear behavior can make correlation missleading.

To further explore the potential associations between land cover and flying fox counts, we plot several combinations of correlated predictors in Figure 13. Zero counts were removed to improve visibility of associations in characteristics of feeding locations. Plots (B), (C), (D), (F), and (G) show clusters of large points away from the edges, providing some evidence of potential interactions. Concluding our exploratory data analysis, we have preliminary evidence that water, flooded areas, trees, shrubs, and developed areas have some association with flying fox ocunts.
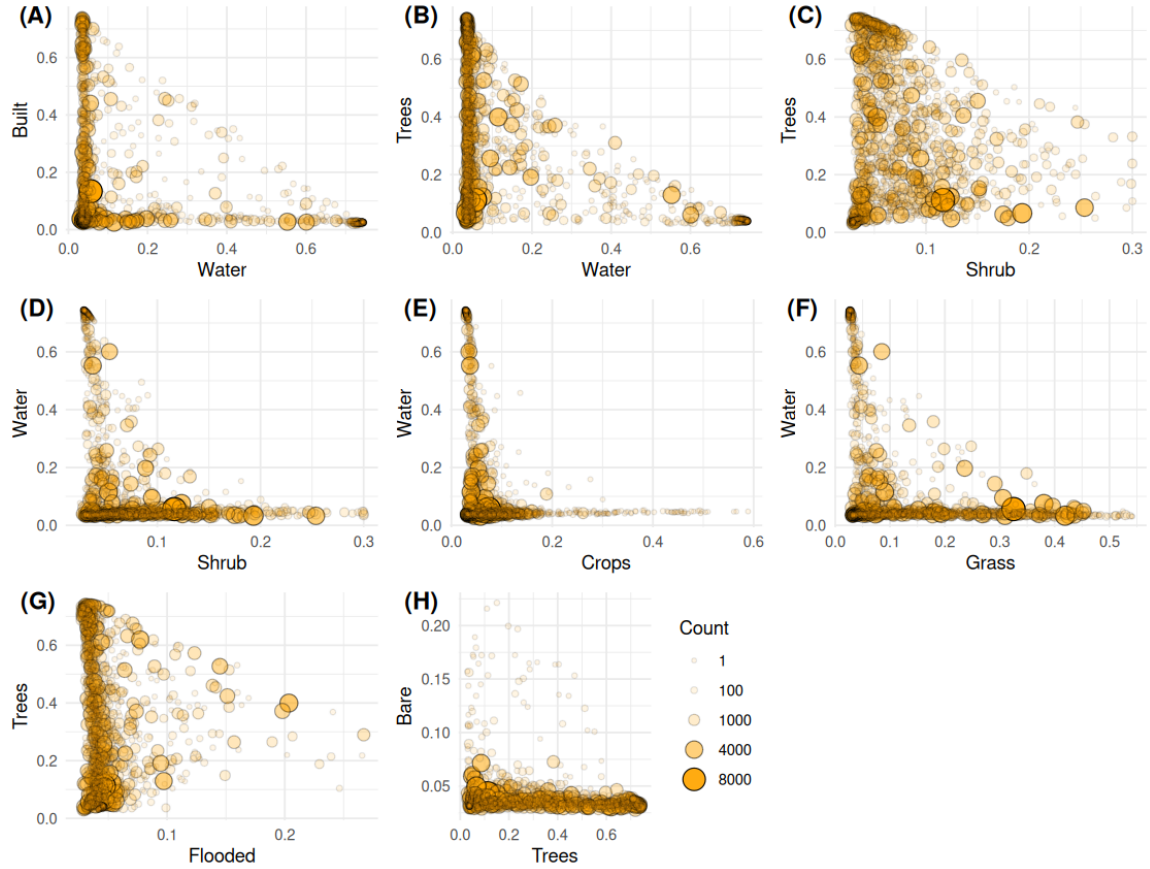
Figure 13: Scatter plots to show potential two-way interactions of land cover classes. Looking towards the center of each plot, large dots indicate a potentially usefult interaction for predictions at the most popular areas. There is some evidence of trees interacting with water, shrubs, and flooded vegation. Water potentially interacts with grass as well.

4.2 Predictions and Diagnostics

Prior to fitting any BART models, we created a 70/30 training/testing split. An initial BART model was fit with default the default parameters and hyperparameters. Convergence was assessed with a trace plot for σ. The burn in period was increased by 50,000 samples at a time until σ had stabilized. The initial model converged reasonably well after a burn in period of 200,000 with 50,000 total

draws. Continuous rank probability scores (CRPS) obtained from the crps_sample function from the scoringRules [Allen, 2024] package are displayed in Figure 14. We see low CRPS values indicating accurate, precise predictions on the right hand side of our area of interest and nearly nowhere else. The model was able to predict that bat's are not likely to feed in the ocean, but not much else. This makes some sense as even if an area is abundant with food, flying foxes may not travel there because resources are available somewhere closer. Results from our initial model indicate land cover alone cannot predict feeding locations.
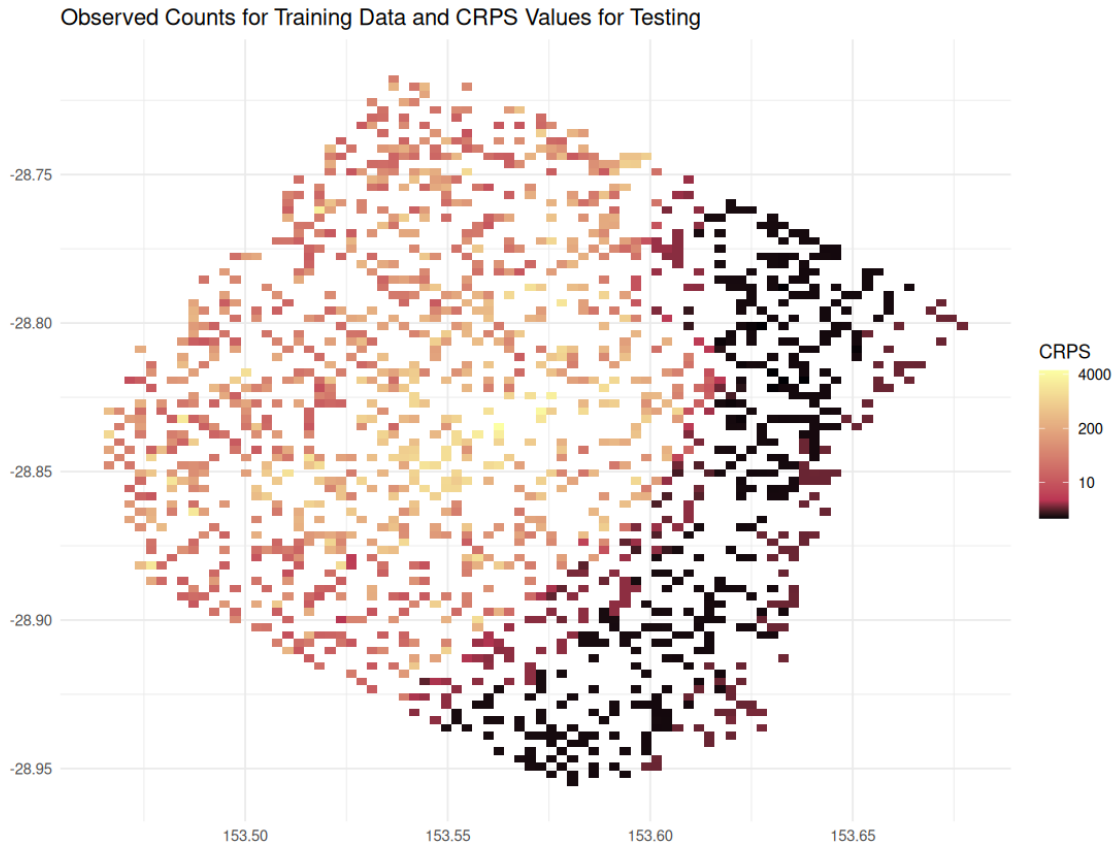

Figure 14: We see the coastline and everywhere to the left has fairly light colors, indicating poor predictions. Without considering the energy required to travel to the edges of our area of intest, the model struggles to discern what land characteristics flying foxes are drawn to.

We need some way to account for the additional energy expended in travelling to the edges of the region of interest. Using the National Flying Fox Monitor Viewer [Australian Government Department of Climate Change, Energy, the Environment and Water, n.d.], we obtained coordinates for the largest flying fox camp within our observed area and calculated the distance to each of our grid cells. Other camps within the region are smaller and located nearby, suggesting additional distance to camp covariates would add little information. With the distance to camp as a covariate, we refit our BART model. A trace plot showing σ has converged reasonably well is displayed in Figure 15. The trace shows some variation in the vertical position of the band of samples, but after 200,000 samples, there is no clear trend. The newly fitted BART model was then evaluated on the testing data.
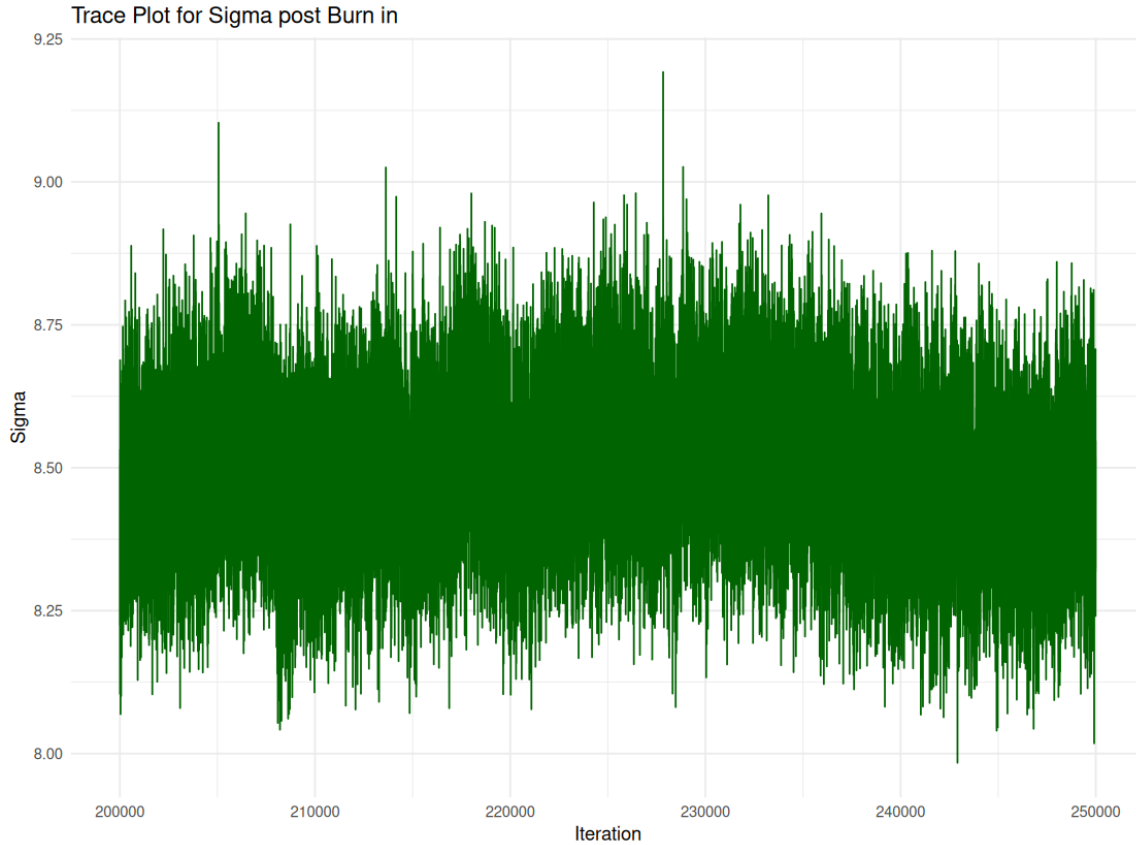
Figure 15: Earlier samples tended to decrease with iterations of MCMC, so a large
burn in period was used. Even with some variation in the green band
of samples, 50,000 draws should give a posterior that is characterized
reasonably well for inference. Our trace plot for σ gives evidence the model
has converged, but does not guarantee the posterior predictive distribution
has converged at each site. This is unlikely to impact inference and was
thus ignored.

Using the crps_sample function, we evaluated our model that accounts for

distance to the camp. Figure 16 displays the CRPS values at testing sites along

with the observed counts for training data. Again, our model's best predictions

essentially say flying foxes will not disappear into the ocean. While the model

clearly struggles to predict the highest flying fox counts, we see darker cells within

the bright blue blob where the majority of flying foxes feed. This suggests BART

was able to learn some features associated with foraging locations. With such a large range of CRPS values, it's difficult to estimate values visually. A table of summary statistics for the CRPS values is displayed in Table 1.
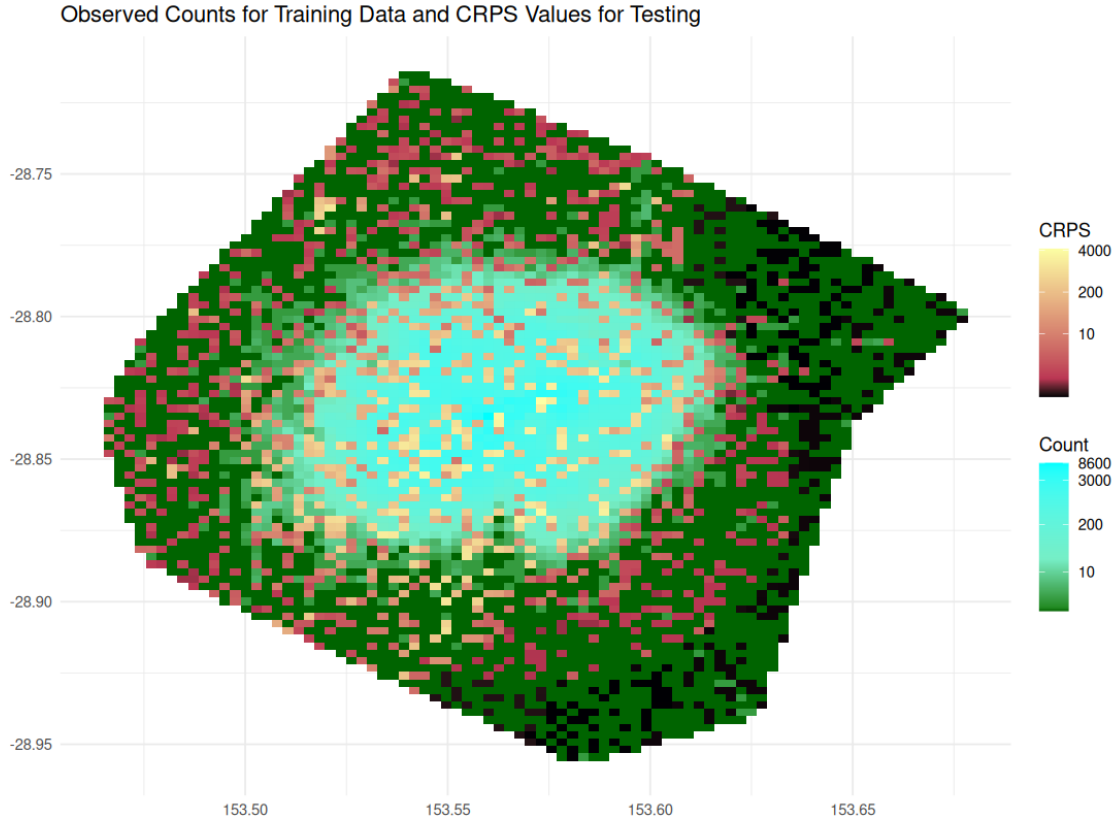

Figure 16: The CRPS scores along with observed counts show our model has some ability to predict flying fox feeding locations out of sample. Model misspecification may exacerbate high CRPS values at larger counts. In count distributions, the variance increases with the mean. This makes violations of the assumptions of constant variance and normally distributed errors probable.

Table 1: Summary statistics for CRPS values. While some predictions are far off, most are reasonable. About 60% of grid cells have zero counts, suggesting the model is good at identifying where flying foxes don't feed.

min	Q1	median	Q3	max	mean	σ
0.108	0.39	1.235	36.377	4463.611	127.127	356.402

To further assess our models ability to predict foraging sites, we present histograms of the observed and predicted counts in Figure 17. Histograms were chosen because we treat counts as continuous. Our BART model predicts large negative values for several locations. Log plus one and log plus ϵ (ϵ small) transformation for the response were considered, but presented there own issues. The log plus one transformation exacerbated poor predictions at sites with large counts. The log plus ϵ transformation makes zero counts very distinct from single counts. With our goal of discerning feeding locations, the response was left untransformed. For nonzero counts, the posterior predictive distribution is reasonably close to the observed counts. Generally, our model underestimates the number of counts within each bin. A truncated BART model would likely better reflect the data, as the probability of counts below zero would be added to observable values. Some of our larger CRPS values can be attributed to model misspecification.
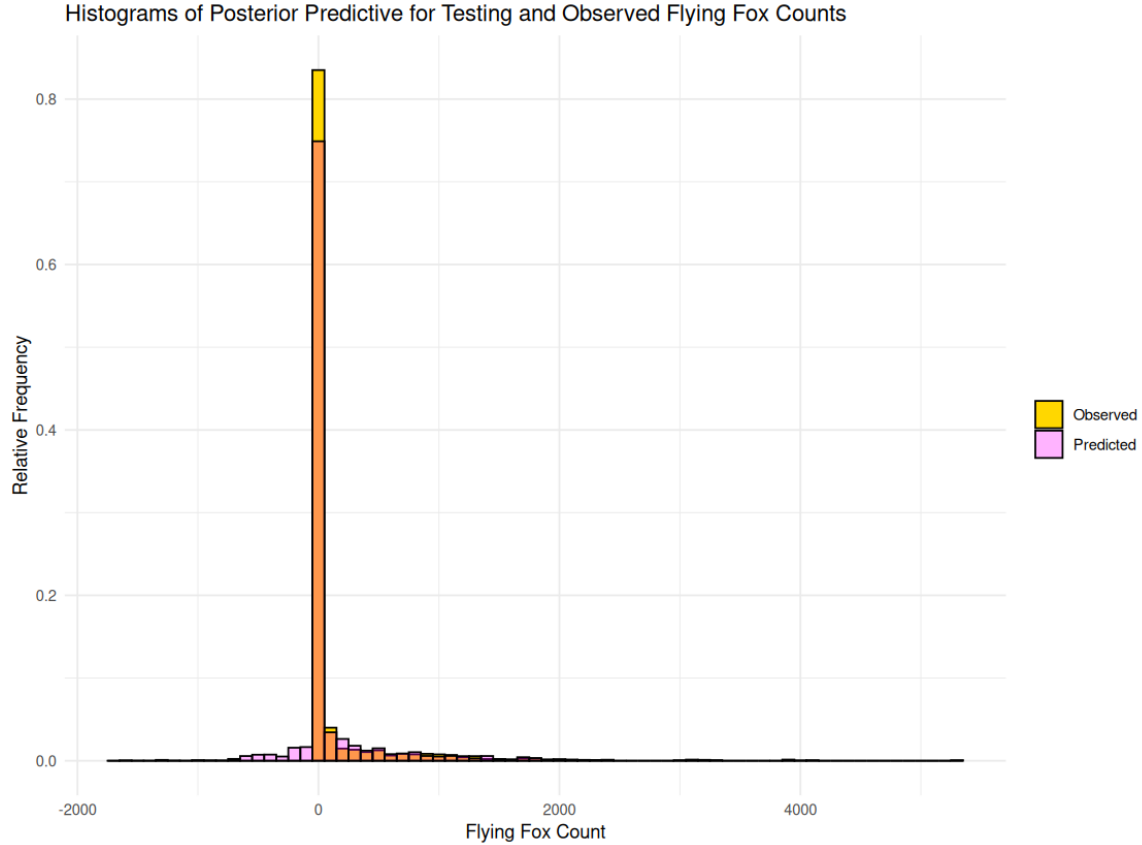
Figure 17: Other than the large left tail below zero, the histograms of the posterior predictive distribution for the testing data and observed counts are fairly similiar in shape. This indicates BART makes mostly reasonable predictions, but does not say anything about the predictions at individual sites.

After assessing our models predictive capabilities, we checked model diagnostics with the Q-Q and residual vs fitted values plots displayed in Figure 18. Mean predictions were used instead of the full posterior for computational efficiency. The Q-Q plot shows very heavy tails, indicating a severe violation of our assumption of normality. In the residuals vs fitted values plot, we see a distinct slanted line of residuals due to counts being non-negative. This creates issues with the assumption of a linear relationship between μ_y as large negative predictions create strictly

positive residuals. This in turn forces overestimation of larger predicted values as residuals are assumed to have mean zero. Our assumption of constant variance appears fairly reasonable as the vertical spread of residuals is fairly consistent. Overall, model diagnostics appear very poor, decreasing our confidence in inferences drawn from the model.

(A) Quantile-Quantile Plot for Residuals from Mean BART Predictions on Training

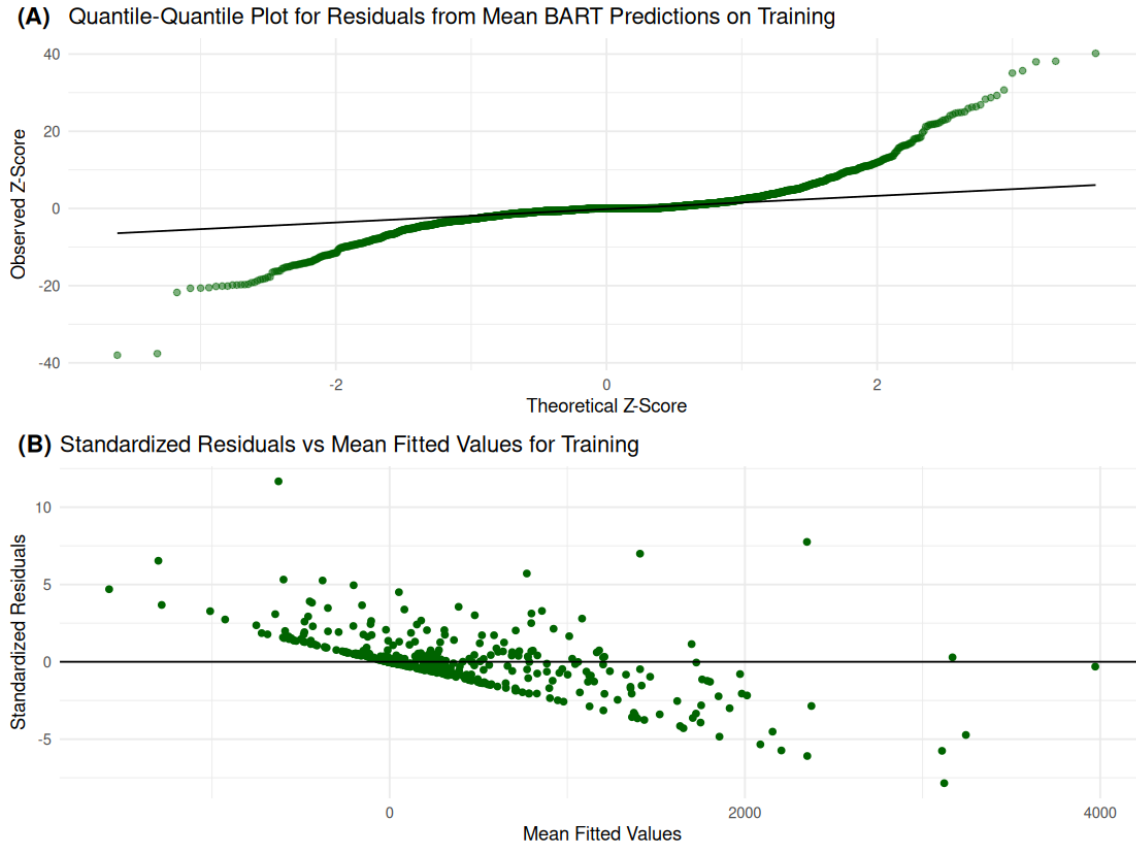**(B)** Standardized Residuals vs Mean Fitted Values for Training

Figure 18: Our residual diagnostic plots show clear violations of normality and linearity. Even without plotting the full posterior residuals, we can tell our model is miscallibrated. Further results must be taken with a grain of salt.

Keeping our models limitations in mind, we present a series of violin plots for variable inclusion probabilities in Figure 19. Distance to the flying fox camp is by far the most important factor. Intuitively this makes sense as animals tend to find

the path of least resistance. Human developed areas (built), shrub, and tree land cover classes display the next highest inclusion probabilities. We can infer a positive association with flying fox counts and trees as fruit bats forage in trees, and the radar cannot track objects below tree canopy. Flooded vegetation, grass, and water also have relatively high inclusion probabilities. The Bare land and crop classes display relatively low variable inclusion probabilities, but the distributions do not include zero implying they were consistently used. The Dynamic World bare and crop classes display fairly low probabilities across the area of interest, so it makes sense that they will be used in fewer decisions. Variable inclusion probabilities are a purely relative measure, so it is difficult to say whether any of the classes are truly associated with foraging sites. Confounding variables are certainly a concern but we can draw some faith from the fact that a class we expect to have some association (trees), displays a similar inclusion probability to the majority of the other classes.
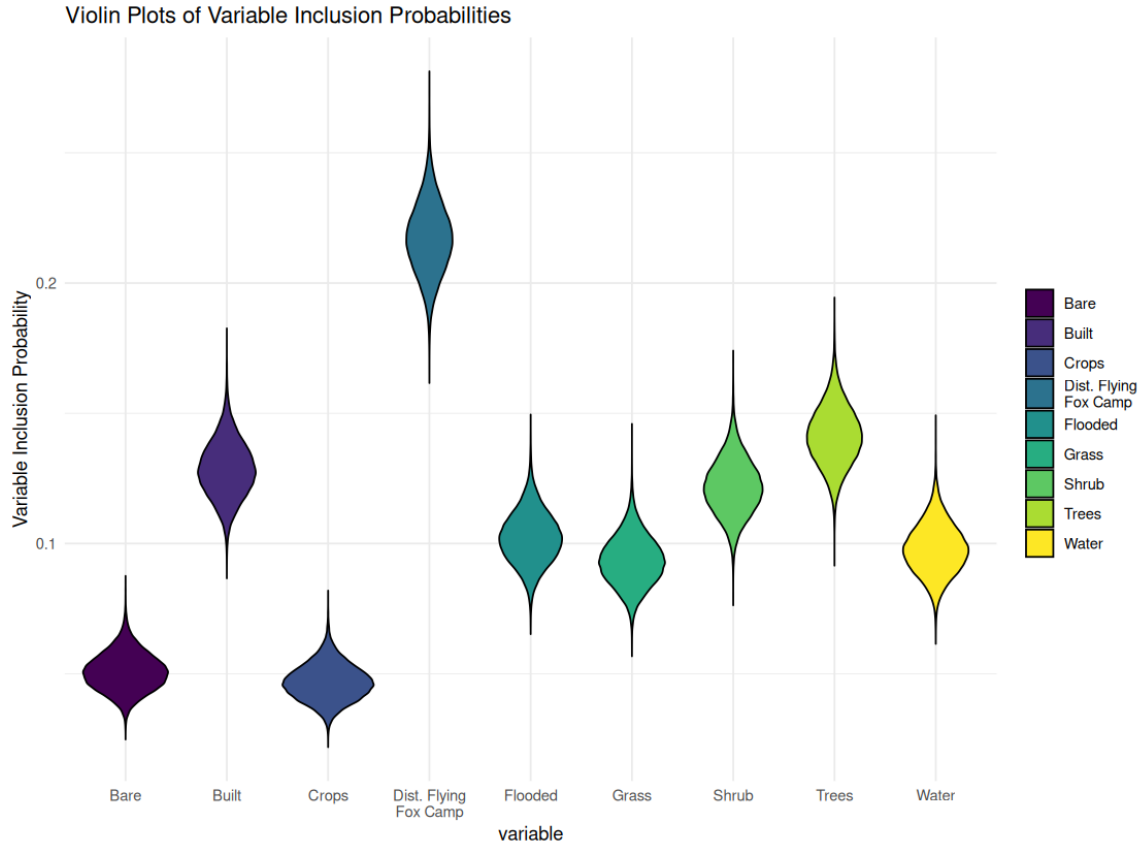
Figure 19: Violin plots of variable inclusion probabilities show the distance to the flying fox camp is most important. Bare land and crops play the least significant role while all other variables have fairly similiar inclusion probabilities. We expect trees to have some association based on background knowledge. Similiar inclusion probabilities for other classes provide some evidence of associations with flying fox counts.

5 Conclusion

Using BART to estimated the flying fox count for three nights of data in Ballina

Australia and found some evidence of associations between feeding locations and

trees, water, shrubs, grass, flooded vegetation, and man made areas. Poor

predictions in high density feeding areas along with violations of modelling

assumptions indicate all results are subject to heavy scrutiny. Our model wasn't capable of explaining why so many flying foxes feed in the center of our area of interest, but it appears to have identified some of the characteristics associated with the area. We can't infer causality from any of the potential associations identified as this is an observational study. All results should be taken with a grain of salt.

It's unlikely that our nine covariates are the only factors that influence flying fox feeding locations. Other covariates such as additional land cover characteristics appear to have a non-negligible role as our residuals showed non-white noise. Thinking about land cover, detailed classes likely provide important information. For example, flying foxes are a type of fruit bat, indicating they will be drawn to fruit bearing trees. Exploring additional land cover classes and using models that constrain the response to be non-negative are likely to improve prediction accuracy and precision. Finally, it may also be worthwhile to censor estimated land class probabilities below a certain threshold as low class probabilities may be purely noise. It's possible that land cover alone can provide a reasonable explanation of where flying foxes feed, but a larger variety of classes is certainly required.

References

Sam Allen. Weighted scoringRules: Emphasizing particular outcomes when evaluating probabilistic forecasts. *Journal of Statistical Software*, 110(8):1–26, 2024. doi: 10.18637/jss.v110.i08.

Australian Government Department of Climate Change, Energy, the Environment and Water. National flying-fox monitoring viewer, n.d. URL https://www.environment.gov.au/webgis-framework/apps/ffc-wide/ffc-wide.jsf. Accessed: 2025-05-09.

Kelsee Baranowski and Nita Bharti. Habitat loss for black flying foxes and implications for hendra virus. *Landscape Ecology*, 38(6):1605–1618, Apr 2023. doi: 10.1007/s10980-023-01642-w.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996. doi: 10.1007/bf00058655.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/a:1010933404324.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and regression trees*, 1984. doi: 10.1201/9781315139470.

Christopher F. Brown, Steven P. Brumby, Brookie Guzder-Williams, Tanya Birch, Samantha Brooks Hyde, Joseph Mazzariello, Wanda Czerwinski, Valerie J. Pasquarella, Robert Haertel, Simon Ilyushchenko, and et al. Dynamic world, near

real-time global 10 m land use land cover mapping. *Scientific Data*, 9(1), Jun 2022. doi: 10.1038/s41597-022-01307-4.

Hugh Chipman, Edward I. George, and Robert E. McCulloch. A Bayesian approach to CART. In David Madigan and Padhraic Smyth, editors, *Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics*, volume R1 of *Proceedings of Machine Learning Research*, pages 91–102. PMLR, 04–07 Jan 1997. URL https://proceedings.mlr.press/r1/chipman97a.html. Reissued by PMLR on 30 March 2021.

Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1), Mar 2010. doi: 10.1214/09-aoas285.

Peggy Eby, Alison J. Peel, Andrew Hoegh, Wyatt Madden, John R. Giles, Peter J. Hudson, and Raina K. Plowright. Pathogen spillover driven by rapid changes in bat ecology. *Nature*, 613(7943):340–344, Nov 2022. doi: 10.1038/s41586-022-05506-2.

Noel Gorelick, Matt Hancher, Mike Dixon, Simon Ilyushchenko, David Thau, and Rebecca Moore. Google earth engine: Planetary-scale geospatial analysis for everyone. *Remote sensing of environment*, 202:18–27, 2017.

Mark Greenwood. *catstats2: Upper Level Statistics for Montana State University Bobcats*, 2025. URL https://github.com/greenwood-stat/catstats2. R package version 0.3.0, commit 23d2979c6bb91f403e5eb27296962654a3e44224.

Trevor Hastie, Robert Tibshirani, and J. H. Friedman. *The elements of Statistical Learning: Data Mining, Inference, and prediction.* Springer, 2 edition, 2009.

Robert J. Hijmans. *raster: Geographic Data Analysis and Modeling*, 2025. URL https://CRAN.R-project.org/package=raster. R package version 3.6-31.

Tim Kam Ho. Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1:278–282, Aug 1995. doi: 10.1109/icdar.1995.598994.

Alan Inglis, Andrew Parnell, and Catherine Hurley. *bartMan: Create Visualisations for BART Models*, 2024. URL https://CRAN.R-project.org/package=bartMan. R package version 0.1.1.

Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002. URL https://CRAN.R-project.org/doc/Rnews/.

Edzer Pebesma. Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal*, 10(1):439–446, 2018. doi: 10.32614/RJ-2018-009. URL https://doi.org/10.32614/RJ-2018-009.

Kanad Roy, Goutam Kumar Saha, and Subhendu Mazumdar. How free-ranging indian flying fox (pteropus medius) forage in urban areas? a study from kolkata, india. *Journal of Urban Ecology*, 10(1), Feb 2024. doi: 10.1093/jue/juae007.

Rodney Sparapani, Charles Spanbauer, and Robert McCulloch. Nonparametric machine learning and efficient computation with Bayesian additive regression

trees: The BART R package. *Journal of Statistical Software*, 97(1):1–66, 2021. doi: 10.18637/jss.v097.i01.

Terry Therneau and Beth Atkinson. *rpart: Recursive Partitioning and Regression Trees*, 2025. URL https://CRAN.R-project.org/package=rpart. R package version 4.1.24.