An explicit split point procedure in model-based trees allowing for a quick fitting of GLM trees and GLM forests

Christophe Dutang, Quentin Guibert

An explicit split point procedure in model-based trees allowing for a quick fitting of GLM trees and GLM forests

Christophe Dutang*, Quentin Guibert*,†

November 25, 2021

Abstract

Classification and regression trees (CART) prove to be a true alternative to full parametric models such as linear models (LM) and generalized linear models (GLM). Although CART suffer from a biased variable selection issue, they are commonly applied to various topics and used for tree ensembles and random forests because of their simplicity and computation speed. Conditional inference trees and model-based trees algorithms for which variable selection is tackled via fluctuation tests are known to give more accurate and interpretable results than CART, but yield longer computation times. Using a closed-form maximum likelihood estimator for GLM, this paper proposes a split point procedure based on the explicit likelihood in order to save time when searching for the best split for a given splitting variable. A simulation study for non-Gaussian response is performed to assess the computational gain when building GLM trees. We also propose a benchmark on simulated and empirical datasets of GLM trees against CART, conditional inference trees and LM trees in order to identify situations where GLM trees are efficient. This approach is extended to multiway split trees and log-transformed distributions. Making GLM trees possible through a new split point procedure allows us to investigate the use of GLM in ensemble methods. We propose a numerical comparison of GLM forests against other random forest-type approaches. Our simulation analyses show cases where GLM forests are good challengers to random forests.

keywords: GLM, model-based recursive partitioning, GLM trees, random forest, GLM forest

1 Introduction

Recursive binary partitioning is a statistical technique for building decision trees by separating a dataset into different homogeneous subgroups according to partitioning explanatory variables that characterize them generally known as features. These models have been widely used in supervised learning for regression and classification for more than 50 years (Loh, 2014). Most binary trees comprise two steps: (1) recursively splitting the dataset by selecting the best split, which forms a node or a leaf, until reaching a stopping criterion; (2) fitting an intercept-only model at each terminal node. CART (Breiman et al., 1984) is certainly the most used model because it is efficient and easy to interpret. However, it suffers from a well-known selection bias problem induced by an exhaustive search over all possible splits simultaneously, see, e.g., Hothorn et al. (2006) or Breiman et al. (1984, p. 42). Several solutions permitting an unbiased split selection in tree algorithms based on statistical tests have been proposed in the literature, see, e.g., FACT (Loh and Vanichsetakul, 1988), QUEST (Loh and Shih, 1997), CRUISE (Kim and Loh, 2001) or CTREE (Hothorn et al., 2006) algorithms. However, these approaches may lead to long computation times, especially on large datasets.

While these classical algorithms are built on a set of nodes with constant values for predictions, the concept of decision trees can be cleverly combined with parametric models through MOdel-Based recursive partitioning (MOB) introduced by Zeileis et al. (2008). The idea consists in fitting separate parametric models for each terminal node of a tree with specific subgroup parameters. For instance, Rusch and Zeileis (2013) and Seibold et al. (2018) use the MOB approach with LM trees and GLM trees. For growing the tree, a unique objective function, such as the log-likelihood function, is considered both

*CEREMADE, Univ. Paris-Dauphine, Univ. PSL, CNRS, Pl. du Ml de Lattre de Tassigny, F-75016 Paris
†Prim'Act, 42 av. de la Grande Armée, F-75017 Paris

for the model parameter estimation and partitioning variables selection. These variables are selected via a fluctuation test (Zeileis and Hornik, 2007) for parameter instability, which overcomes the variable selection-bias issue. This general framework is the successor of different previous models which also integrate a parametric model into a tree, see among others *generalized regression trees* (Ciampi, 1991), *functional trees* (Gama, 2004), GUIDE (Loh, 2002) or *maximum likelihood trees* under the assumption of a Gaussian response (Su et al., 2004). Using a well-established parametric model, a key goal of MOB is that it provides predictions given by a statistical model adapted to each node, which are easy to interpret. However, the computation time may be longer than simple tree algorithms with intercept-only nodes since the splitting procedure requires maximizing the objective function and assessing the parameters instability using a hypothesis test.

For all tree methods, single trees suffer from an instability issue, i.e., the resulting tree can be significantly affected by small changes in the training data. See Philipp et al. (2018) for a general framework for assessing the stability of results generated from supervised statistical learning algorithms. They may be less competitive than other classical approaches in artificial intelligence and machine learning, like neural networks (Lawrence, 1994) or support vector machines (Cortes and Vapnik, 1995) in terms of prediction. Predictions can be improved by introducing ensemble tree methods based on bagging (Breiman, 1996), random forest (Breiman, 2001) or boosting (Friedman, 2002), but this is done at the expense of interpretability. In this literature, the overwhelming majority of approaches are based on the CART algorithm, although variable selection may be biased. Unbiased trees for forest are also available, see function `cforest` in (Hothorn and Zeileis, 2015) for the CTREE algorithm introduced in Hothorn et al. (2006) or the function `mobForest` related to random forest for MOB (Garge et al., 2013), both implemented in R (R Core Team, 2021). Since ensembles are based on random sampling, the longer computation time of these unbiased tree methods makes them more difficult to use than ensemble trees based on CART, as stated, e.g., by Garge et al. (2013) or Fokkema (2020).

In this paper, we propose a new approach for reducing the computation time of recursive binary partitioning based on the explicit fitted GLM likelihood as an objective function. Based on the idea of Brouste et al. (2020), we show that closed-form estimators can be derived when computing the split points with respect to each partitioning variable for GLM-type trees. More precisely, we replace a time-consuming step in the splitting procedure where the split point is determined by optimizing a score function based on the GLM likelihood. This approach is general as the results obtained by Brouste et al. (2020) are valid for any distribution belonging to the one-parameter exponential family (possibly with an additional dispersion parameter) and any link function in the case of categorical explanatory variables. Hence, it can be applied to several GLM-type models such as maximum likelihood trees (Su et al., 2004) or GLM Trees (Rusch and Zeileis, 2013). We specifically focus on this last approach in that paper due to its flexibility.

The remainder of this paper is organized as follows. In Section 2, we present GLMs and GLM trees as well as the framework for building GLM trees based on a closed-form estimator. Section 3 assesses the performances of our approach on simulated and empirical datasets. In Section 4, we propose a GLM forest algorithm based on GLM trees based on bagging. Finally, Section 5 concludes.

2 Generalized Linear Models Binary Trees

In this section, we first describe GLMs and GLM-based trees and explain how we introduce a closed-form estimator for GLMs in the original GLM tree algorithms. Then, we discuss how this approach can be extended to other decision trees based on Maximum Likelihood Estimation (MLE). Finally, several improvements of estimators proposed by Brouste et al. (2020) and useful for tree models are introduced.

In the following, for the sake of clarity, we use bold notations for vectors of $\mathbb{R}^p$, $\mathbb{R}^q$ or $\mathbb{R}^n$, where $p, q, n \in \mathbb{N}^\star$. We use the letter x for explanatory variables on which GLMs are fitted and the letter z for partitioning variables used for the split point procedure. The index $i \in I = \{1, \ldots, n\}$ is reserved for the observations, while the indexes j, k, l are used both for explanatory and partitioning variables.

2.1 Generalized linear models

Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ be deterministic exogenous explanatory variables, with $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,p}) \in \mathbb{R}^p$ for $i \in I$. It is assumed that $x_{i,j}$ is either a real for numeric variables or a binary dummy for categorical variables. Generalized linear models, see, e.g., McCullagh and Nelder (1989), assume that (i) the random response vector $\boldsymbol{Y}$ has independent components, (ii) the distribution of random variable Y_i, $i \in I$, belongs to the exponential family with natural parameter λ_i, see Equation (1), (iii) a so-called link function g governs the relation between random response variables Y_i and deterministic explanatory variables $\boldsymbol{x}_i$, see Equation (2). In the rest of the paper, lower case y_i is used for the corresponding observation of the random variable Y_i.

In other words, the log-likelihood associated with a single observation y_i of the random variable Y_i, $i \in I$, in the exponential family is given by

$$\log L(\lambda_i, \phi, y_i) = \frac{\lambda_i y_i - b(\lambda_i)}{a(\phi)} + c(y_i, \phi), \quad y_i \in \mathbb{Y} \subset \mathbb{R}, \quad \lambda_i \in \Lambda \subset \mathbb{R}, \tag{1}$$

and $-\infty$ if $y_i \notin \mathbb{Y}$, where $a : \mathbb{R} \to \mathbb{R}$, $b : \Lambda \to \mathbb{R}$ and $c : \mathbb{Y} \times \mathbb{R} \to \mathbb{R}$ are known real-valued differentiable measurable functions. The term λ_i is called the natural parameter and ϕ is the dispersion parameter.

Since the expectation is $E[Y_i] = b'(\lambda_i)$ for the exponential family, the relationship between the expectation of Y_i and $\boldsymbol{x}_i$ is governed by

$$g(b'(\lambda_i(\boldsymbol{\theta}))) = \langle \boldsymbol{x}_i, \boldsymbol{\theta} \rangle = \eta_i, \quad \forall \boldsymbol{\theta} \in \Theta \subset \mathbb{R}^p, \tag{2}$$

where η_i is the linear predictor, $\langle ., . \rangle$ denotes the scalar product, and $\lambda_i(\boldsymbol{\theta})$ stresses the dependence on the finite-dimensional parameter $\boldsymbol{\theta}$. Since g is a bijective function, we have an equivalent relationship $\lambda_i(\boldsymbol{\theta}) = \tilde{b} \circ g^{-1}(\langle \boldsymbol{x}_i, \boldsymbol{\theta} \rangle)$ where $\tilde{b} = (b')^{-1}$ is the inverse of b'. The link function g is called canonical when $g = \tilde{b}$. Some usual examples are given in Table 1.

It is worth noting that $\boldsymbol{\theta}$ is estimated in the first step. Secondly, the dispersion parameter ϕ is estimated if any. This additional dispersion parameter makes possible to use non-constant variance distributions such as gamma or inverse Gaussian distributions, see, e.g., McCullagh and Nelder (1989, Chapter 8).

Distribution	λ	ϕ	$a(x)$	$b(x)$	$c(x, \phi)$	$b'(x)$	$\tilde{b}(x)$	$b(\tilde{b}(x))$
Bernoulli $\mathcal{B}(p)$	$\log(\frac{p}{1-p})$	1	x	$\log(1+e^x)$	0	$\frac{e^x}{1+e^x}$	$\log(\frac{x}{1-x})$	$-\log(1+x)$
Binomial $\mathcal{B}(m,p)$	$\log(\frac{p}{1-p})$	$\frac{1}{m}$	x	$\log(1+e^x)$	$\log\left(\binom{1/\phi}{x/\phi}\right)$	$\frac{e^x}{1+e^x}$	$\log(\frac{x}{1-x})$	$-\log(1+x)$
Gaussian $\mathcal{N}(\mu,\sigma^2)$	μ	σ^2	x	$x^2/2$	x^2/ϕ $-\frac{1}{2}\log(2\pi\phi)$	x	x	$x^2/2$
Gamma $\mathcal{G}(\nu,\mu)$	$\frac{-1}{\mu}$	$1/\nu$	x	$-\log(-x)$	$\frac{\log(x/\phi)}{\phi} - \log(x)$ $-\log(\Gamma(\frac{1}{\phi}))$	$-1/x$	$-1/x$	$\log(x)$
Poisson $\mathcal{P}(\mu)$	$\log(\mu)$	1	x	e^x	$-\log(x!)$	e^x	$\log(x)$	x
Inv. Gauss. $\mathcal{IG}(\mu,\sigma^2)$	$-1/(2\mu^2)$	$1/\sigma^2$	x	$-\sqrt{-2x}$	$-\frac{1}{2}\log(2\pi\phi x^3)$ $-1/(2\phi x)$	$1/\sqrt{-2x}$	$-1/(2x^2)$	$-1/x$

Table 1: Usual distributions in the exponential family

If the model is identifiable, the MLE $\widehat{\boldsymbol{\theta}}_n$ has desirable properties (existence, consistence, asymptotic normality), see Fahrmeir and Kaufmann (1985). In the general case, when $\boldsymbol{x}_i$ is a vector of numerical and/or categorical explanatory variables, there is no explicit solution for the MLE and estimation relies on an iterative weighted least-square (IWLS) algorithm. Conversely, it is well known that the MLE can be derived directly with a closed-form formula with the Gaussian family, see e.g. Seber and Lee (2003) and Weisberg (2005).

2.2 GLM-based trees and explicit likelihood split point

The GLM-based tree algorithm introduced by Rusch and Zeileis (2013) consists of splitting the dataset recursively based on a set of partitioning variables and of fitting a GLM on a set of explanatory variables

to observations in each node. First, the selection of the variable $j^\star$ among all partitioning variables is performed by assessing the parameter stability and by finding the partitioning variable with the highest parameter instability. Given the partitioning variable $j^\star$, this procedure locally estimates in a second step a parametric model for all possible splits of that variable, see Section 2.2.1. This step can be time-consuming since an exhaustive search is realized is performed to find the optimal split point $s^\star$ which requires to fit as many GLM as possible split points. For instance for a categorical variable x with 4 levels A, B, C, D, the following six GLMs are fitted using the IWLS algorithm:

- left node $1_{x\in\{A,B,C\}}$ against right node $1_{x\in\{D\}}$,
- left node $1_{x\in\{A,B,D\}}$ against right node $1_{x\in\{C\}}$,
- left node $1_{x\in\{A,C,D\}}$ against right node $1_{x\in\{B\}}$,
- left node $1_{x\in\{A,B\}}$ against right node $1_{x\in\{C,D\}}$,
- left node $1_{x\in\{A,C\}}$ against right node $1_{x\in\{B,D\}}$,
- left node $1_{x\in\{A,D\}}$ against right node $1_{x\in\{B,C\}}$.

For these two steps, we show in Section 2.2.2 how to improve this procedure based on the results of Brouste et al. (2020), where a closed-form solution for GLMs with categorical variables is available for any link function g and for any probability distribution. More precisely, explicit solutions are derived for GLM-based tree models for any type of partitioning variables in situations where the GLMs are fitted with only an intercept or with one categorical explanatory variable. The difference between non-explicit and explicit GLM trees is how GLMs are fitted: the first one relies on the IWLS algorithm to compute the MLE, while our procedure computes the closed-form MLE when the tree is growing.

2.2.1 Model-based partitioning trees

The MOB model introduced by Zeileis et al. (2008) is a flexible framework of model parameter estimation based on least squares, maximum likelihood or more broadly M-estimation approaches. Within this approach, the parametric model $\mathcal{M}_{\mathcal{B}_b}\left(\boldsymbol{Y}, \{\boldsymbol{x}_i\}, \{\lambda\}\right)$, $b = 1, \ldots, B$, $i \in I$, is locally fitted using vectors of explanatory covariates $\{\boldsymbol{x}_i\}$ on each subset of a partition $\{\mathcal{B}_b\}$ of B segments. The partition is determined when growing a tree based on partitioning variables $\boldsymbol{z}_i = (z_{i,1}, \ldots, z_{i,q}) \in \mathbb{R}^q$, and an objective function is maximized to obtain the generic collection of parameters $\{\lambda_b\}$.

In this paper, the local model on which we focus is the GLM similarly to Rusch and Zeileis (2013) where parameters $\boldsymbol{\theta}_b$, $b = 1, \ldots B$, are estimated by maximizing the log-likelihood on the set of explanatory covariables. Their algorithm for GLM trees is given in Algorithm 1. Contrary to CART, MOB doesn't require a post-pruning procedure of the tree. For instance, a pre-pruning step can be applied to avoid the size of a node becomes too small.

First, a full GLM is fitted on all observations of the current node b ($i \in b$) with all explanatory variables available $\boldsymbol{x}_i$ (continuous and/or categorical). Candidate variables for splitting are either all partitioning variables or a subset of them if a random selection is performed. Unless explanatory variables come from a single categorical variable, for which an explicit MLE exists (Brouste et al., 2020), $\hat{\boldsymbol{\theta}}_b$ is computed by the IWLS algorithm.

Second, a variable selection is performed to avoid selecting useless variables. It is based on a M-fluctuation test[1]. Let $W_j(t)$ be an empirical fluctuation process, i.e., a normalized cumulative score process. Zeileis and Hornik (2007) show that under regularity conditions, a normalized $W_j(t)$ converges toward a standard Brownian bridge, which allows to perform M-fluctuation tests for testing the null hypothesis of parameter stability for each partitioning variable and thus to select the most significant one. For GLMs, the i-th score contribution is explicit and given by

$$s_{i,j}(\boldsymbol{\theta}) = \frac{\partial \log L(\lambda_i, y_i)}{\partial \beta_j} = \frac{y_i - \mu_i}{V(\mu_i)} h'(\eta_i) z_{i,j}, \tag{3}$$

[1] Algorithm 1 assesses parameter instability only if the splitting variable has two or more levels for a categorical variable, or two or more values for a numeric variable. Otherwise, there is nothing to split.

Data: Response vector $\boldsymbol{y}$, vector of explanatory covariates $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, partitioning variables $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$

while *Loop over node b until no significant instability is detected* **do**

Compute the observation number $n_b = \sharp b$ for node b.

if *n_b is too small* **then**

Stop the process for that node.

end

1. Fit the local model:

Fit GLM by maximizing (1) for obs. $i \in b$ to obtain fitted parameter $\hat{\boldsymbol{\theta}}_b$.

2. Assess param. instab. with M-fluctuation tests:

for $j = 1, \ldots, q$ **do**

Compute the i-th score contribution as $\hat{s}_{i,j} = s_{i,j}(\hat{\boldsymbol{\theta}}_b)$ in (3) for all $i \in b$.

if *j is a numerical variable* **then**

Compute parameter instability using (4) as

$$\lambda_j = \max_{i=\underline{i},\ldots,\bar{i}} \frac{(n_b)^2}{i(n_b - i)} \left\| W_j\left(i/n_b, \hat{\boldsymbol{\theta}}_b\right) \right\|_2^2,$$

where $[\underline{i}, \bar{i}]$ is the interval of potential instability.

else

Compute parameter instability using (4) as

$$\lambda_j = \frac{1}{n_b} \sum_{c=1}^{l_j} (\sharp I_{v_{j,c}})^{-1} \left\| \Delta_{v_{j,c}} W_j\left(i/n_b, \hat{\boldsymbol{\theta}}_b\right) \right\|_2^2,$$

where $I_{v_{j,c}} = \{i \in b, z_{i,j} = v_{j,c}\}$ is the set of observation indices in category $v_{j,c}$.

end

end

Compute the p-value of the fluctuation test and assess the significance.

if *there is at least one significant instable variable* **then**

Select the most unstable variable

$$j^\star = \arg\max_{j \in \{1,\ldots,q\}} \lambda_j.$$

3. Choose the best splitting point s:

if *$j^\star$ is a numerical variable* **then**

Search for the optimal split point $s^\star \in (\min_i z_{i,j^\star}, \max_i z_{i,j^\star})$ based on (10).

else

Search for the optimal set $s^\star \subset \{v_{j^\star,1}, \ldots, v_{j^\star,l_j}\}$ based on (10).

end

end

end

Algorithm 1: Recursive partition algorithm for GLMs for a binary tree

where $\mu_i = h(\eta_i)$, $h = g^{-1}$ the inverse link function, $\eta_i = \langle \boldsymbol{x}_i, \boldsymbol{\theta} \rangle$. The corresponding covariance matrix $J(\boldsymbol{\theta})$ is also explicit, see, e.g., Zeileis and Hornik (2007, Section 4.1). Therefore, the empirical fluctuation process has the following expression for GLMs

$$W_j(t, \hat{\boldsymbol{\theta}}) = \hat{J}^{-1/2} \frac{1}{\sqrt{n_b}} \sum_{i \leqslant \lfloor t \times \sharp b \rfloor}^{i \in b} \hat{s}_{\sigma(i),j}, 0 \leqslant t \leqslant 1, \tag{4}$$

where $\hat{s}_{i,j} = s_{i,j}(\hat{\boldsymbol{\theta}})$, $\sigma(i)$ is the ordering permutation giving the anti-rank observation of $z_{i,j}$, $\sharp b$ is the cardinality of the set b and $\hat{J} = J(\hat{\boldsymbol{\theta}})$ the fitted covariance matrix. The function $t \mapsto W_j(t, \hat{\boldsymbol{\theta}})$ is a

step function and we denote the increment for variable j in category v by $\Delta_v W_j(t, \hat{\boldsymbol{\theta}})$. Hence, the splitting variable with the highest significant instability is selected, i.e., the lowest p-value satisfying the significance level adjusted with the Bonferroni correction.

Once the best splitting variable is selected, the algorithm chooses the best split point based on the objective function calculated on the $B \geqslant 2$ daughter nodes of the splitting variable

$$O\left(\boldsymbol{y}, \phi, \hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_B\right) = \sum_{b=1}^{B} \log L(\hat{\boldsymbol{\theta}}_b, \phi, y_i) 1_{\{i \in L_b(j^\star)\}}, \tag{5}$$

where $L_b(j^\star)$ corresponds to the b-th segment with respect to values taken by the variable $j^\star$. For binary tree ($B = 2$), only one split point for a continuous variable or one subset for a categorical variable, hereafter noted s, should be exhibited.

In general, objective function (5) is not explicit because parameter vector $\hat{\boldsymbol{\theta}}_b$ has to be estimated by the IWLS algorithm. In what follows, we focus on this last step and propose closed-form formula for $\left(\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_B\right)$ leading to an explicit objective function. More precisely, our GLM-based tree algorithm with closed-form solutions covers these particular situations:

- a GLM with no explanatory variable and a set of partitioning variables (continuous and/or categorical), see Section 2.2.2.
- a GLM with a single categorical explanatory variable and a set of partitioning variables (continuous and/or categorical), see Section 2.4.2.

When continuous explanatory variables or more than one categorical explanatory variable are considered, it is appropriate to return to the general framework set by Zeileis et al. (2008).

2.2.2 Explicit likelihood split point for binary trees with intercept-only nodes

Based on Brouste et al. (2020), we now detail how exactly the split point can be performed more efficiently for a GLM-based recursive partition, either for a numerical or categorical splitting variable $j^\star$. For the sake of clarity, we present the case of binary trees with intercept-only nodes, i.e., a GLM-based tree with any explanatory variable. Such a tree with intercept-only nodes is especially relevant for comparisons with classical partitioning recursive like the CART or the CTREE algorithms, or for generating ensemble of decisions trees, see, e.g., Fokkema (2020). This presentation can be extended to situations with trees of non-constant nodes, i.e., multiple categorical explanatory variables, and to multiway splits, see Section 2.4.

When the $j^\star$-th partitioning variable is numerical, we consider two binary dummies[2] for a known split point s for all $i \in I$

$$\eta_i = \theta_L \times 1_{\{z_{i,j^\star} \leqslant s\}} + \theta_R \times 1_{\{z_{i,j^\star} > s\}}. \tag{6}$$

Split point values s are taken in the range of the $j^\star$-th partitioning variable: $s \in (\min_i z_{i,j^\star}, \max_i z_{i,j^\star})$. When the $j^\star$-th variable is categorical taking its values in $\{v_{j^\star,1}, \ldots, v_{j^\star,l_j}\}$, we consider two binary dummies for a known subset s for all $i \in I$

$$\eta_i = \theta_L \times 1_{\{z_{i,j^\star} \in s\}} + \theta_R \times 1_{\{z_{i,j^\star} \notin s\}}. \tag{7}$$

Subset values are taken among the partition of the $j^\star$-th partitioning variable levels $s \subset \{v_{j^\star,1}, \ldots, v_{j^\star,l_{j^\star}}\}$.

In order to unify both situations, we use a generic notation for the linear predictor

$$\eta_i = \theta_L \times 1_{\{i \in L(j^\star,s)\}} + \theta_R \times 1_{\{i \in R(j^\star,s)\}}, \tag{8}$$

[2]Our specification has no intercept since two dummy variables are introduced. This means that neither the left nor the right sub-sample is taken as a reference. Note that a model with one intercept and one dummy variable is equivalent and would simply lead to rewrite Equation (6). However, the fitted log-likelihood is identical with or without intercept, see Corollary 3.1 and Examples 3.1, 3.2 of Brouste et al. (2020).

where $L(j,s)$ and $R(j,s)$ are the children subsets resulting from the split. Equation (6) is obtained with $i \in L(j,s) \Leftrightarrow z_{i,j} \in (-\infty, s]$ and $i \in R(j,s) \Leftrightarrow z_{i,j} \in (s, +\infty)$, while (7) is obtained with $i \in L(j,s) \Leftrightarrow i \in s$ and $i \in R(j,s) \Leftrightarrow i \notin s$.

From Theorem 3.1 of Brouste et al. (2020), given the link function g, the MLE $\widehat{\boldsymbol{\theta}}(s) = (\hat{\theta}_L(s), \hat{\theta}_R(s))$ is given by

$$\hat{\theta}_L(s) = g\left(\overline{y}_{j^\star}^L(s)\right), \ \hat{\theta}_R(s) = g\left(\overline{y}_{j^\star}^R(s)\right), \tag{9}$$

where $\overline{y}_{j^\star}^L(s)$, $\overline{y}_{j^\star}^R(s)$ are average responses respectively for left and right subsets conditional on the $j^\star$ value, see Table 2. As mentioned in Brouste et al. (2020), the fitted parameters do not depend on the distribution (functions a, b, c) but only on the link function g.

	Left node	Right node
Node size	$m_j^L(s) = \sum_{i=1}^n 1_{\{i \in L(j,s)\}}$	$m_j^R(s) = \sum_{i=1}^n 1_{\{i \in R(j,s)\}}$
Average	$\overline{y}_j^L(s) = \frac{1}{m_j^L(s)} \sum_{i=1}^n y_i 1_{\{i \in L(j,s)\}}$	$\overline{y}_j^R(s) = \frac{1}{m_j^R(s)} \sum_{i=1}^n y_i 1_{\{i \in R(j,s)\}}$

Table 2: Notations for conditional node sizes and averages

Furthermore, we can deduce the fitted log-likelihood (5) using Corollary 3.1 of Brouste et al. (2020), see Appendix A.1. Equation (15) in Appendix A.1 has terms which do not depend on s and the same applies for the fitted deviance, see Appendix A.2. Hence, we do not need to estimate ϕ nor to depend on the functions a and c in order to maximize the log-likelihood $\log L_j$ with respect to s, i.e., the application $s \mapsto \log L(\hat{\theta}_L(s), \hat{\theta}_R(s), \boldsymbol{y}, s)$ (or the deviance) can be simplified.

Let $\overline{\boldsymbol{y}_{j^\star}}(s) = (\overline{y}_{j^\star}^L(s), \overline{y}_{j^\star}^R(s))$ be the vector of conditional average responses and $\boldsymbol{m}_{j^\star}(s) = (m_{j^\star}^L(s), m_{j^\star}^R(s))$ be the vector of node sizes. Thanks to Brouste et al. (2020), we maximize the following explicit objective function $s \mapsto O(\overline{\boldsymbol{y}_{j^\star}}(s), \boldsymbol{m}_{j^\star}(s))$ given the splitting variable $j^\star$ in order to find the best split point or the best subset $s^\star$ with

$$\begin{aligned} O(\overline{\boldsymbol{y}_{j^\star}}(s), \boldsymbol{m}_{j^\star}(s)) = \tilde{b}\left(\overline{y}_{j^\star}^L(s)\right) m_{j^\star}^L(s) \overline{y}_{j^\star}^L(s) - b\left(\tilde{b}(\overline{y}_{j^\star}^L(s))\right) m_{j^\star}^L(s) \\ + \tilde{b}\left(\overline{y}_{j^\star}^R(s)\right) m_{j^\star}^R(s) \overline{y}_{j^\star}^R(s) - b\left(\tilde{b}(\overline{y}_{j^\star}^R(s))\right) m_{j^\star}^R(s), \end{aligned} \tag{10}$$

where $\tilde{b} = (b')^{-1}$ is the inverse of b'. Equation (10) is thus linked to the distribution assumption only through the function b but neither a, c, nor the link function g. Hence, the choice of link function has an impact on the fitted parameter $\widehat{\boldsymbol{\theta}}(s)$ but neither on the fitted likelihood, nor the objective function.

As noted by Zeileis et al. (2008), the framework introduced by MOB is quite general and unifies different decision tree models, such as maximum likelihood trees introduced by Su et al. (2004) for regression model with intercept only. This remains true in our approach, which is in line with this framework and allows some classical models to benefit from explicit solutions. The objective function (10) is actually very close to classical situations, namely the entropy function and the residual sum of squares used in the CART algorithm.

Consider a Bernoulli response with $\lambda = \log(p/(1-p))$, for which $\mathbb{Y} = \{0, 1\}$, $\Lambda = \mathbb{R}$. Using Table 1 and ignoring s in frequencies and averages for ease of notation, Equation (10) becomes

$$O(\overline{\boldsymbol{y}_j}, \boldsymbol{m}_j) = m_j^L \left[\overline{y}_j^L \log\left(\overline{y}_j^L\right) + (1 - \overline{y}_j^L) \log\left(1 - \overline{y}_j^L\right)\right] + m_j^R \left[\overline{y}_j^R \log\left(\overline{y}_j^R\right) + (1 - \overline{y}_j^R) \log\left(1 - \overline{y}_j^R\right)\right].$$

This formula is of type $p \log(p) + (1-p)\log(1-p)$ as the entropy function used in classification trees, see Venables and Ripley (2002, p. 255).

Note that if for some splits, the left node or the right node contains only successes or failures, the above objective function has an indefinite form $0 \times \infty$. In practice, the limit of $p \log(p)$ is used, i.e., $p \log(p) \to 0$ as $p \to 0$, or analogously $(1-p)\log(1-p) \to 0$ as $p \to 1$. Hence, the contribution of the problematic node to the objection function is set to zero.

For a Gaussian response with a mean $\mu = \lambda$ and a variance $\sigma^2 = \phi$ for which $\mathbb{Y} = \mathbb{R}$ and $\Lambda = \mathbb{R}$, Equation (10) becomes using Table 1 and ignoring s,

$$O(\overline{\boldsymbol{y}_j}, \boldsymbol{m}_j) = \frac{1}{2}\left(\overline{y}_j^L\right)^2 m_j^L + \frac{1}{2}\left(\overline{y}_j^R\right)^2 m_j^R.$$

Despite the formula looks different than the usual regression tree, this objective function is indeed proportional to the loss in deviance. From Chambers and Hastie (1993, p. 414) with their notation, the loss in deviance is defined as

$$\Delta D = \sum_{i=1}^{n}(y_i - \hat{\mu})^2 - \sum_{i\in L}^{n}(y_i - \hat{\mu}_L)^2 - \sum_{i\in R}^{n}(y_i - \hat{\mu}_R)^2 \propto (\hat{\mu}_L)^2 m_L + (\hat{\mu}_R)^2 m_R.$$

Using $\hat{\mu}_L = \overline{y}_j^L$, $\hat{\mu}_R = \overline{y}_j^R$, $m_L = m_j^L$ and $m_R = m_j^R$ leads to ΔD is proportional to our objective function.

2.3 Non-Gaussian non-binary distributions for GLM trees

An advantage of the GLM framework is the possibility to use any probability distribution in the exponential family: we are not limited to Gaussian distribution for numeric responses and Bernoulli distribution for binary responses, see usual distributions in Table 1. In the insurance field, actuaries and statisticians usually use the Poisson distribution to model claim frequencies and the gamma distribution to model claim severities, see, e.g., Denuit et al. (2019). In biology, there are also several examples of the relevancy of non-Gaussian non-binary distributions, e.g., Szöcs and Schäfer (2015) and Wilson and Grenfell (1997).

2.3.1 Classical distributions for GLMs

The objective function (10) contains many other special cases. We give below typical probability distributions generally used within the GLM framework.

A gamma distribution parametrized by its mean μ and its shape parameter ν is obtained with $\lambda = \frac{-1}{\mu}$, $\phi = 1/\nu$, for which $\mathbb{Y} = (0, +\infty)$, $\Lambda = \mathbb{R}_-$. Using Table 1, we get

$$O(\overline{\boldsymbol{y}_j}, \boldsymbol{m}_j) = -m_j^L \left(1 + \log(\overline{y}_j^L)\right) - m_j^R \left(1 + \log(\overline{y}_j^R)\right).$$

A Poisson distribution with a mean μ is obtained with $\lambda = \log(\mu)$, for which $\mathbb{Y} = \mathbb{N}$ and $\Lambda = \mathbb{R}$. Using Table 1, we get

$$O(\overline{\boldsymbol{y}_j}, \boldsymbol{m}_j) = \left(\log(\overline{y}_j^L) - 1\right) \overline{y}_j^L m_j^L + \left(\log(\overline{y}_j^R) - 1\right) \overline{y}_j^R m_j^R.$$

An Inverse Gaussian distribution parametrized by its mean μ and its shape parameter σ^2 is obtained with $\lambda = -1/(2\mu^2)$, $\phi = 1/\sigma^2$, for which $\mathbb{Y} = (0, +\infty)$ and $\Lambda = (-\infty, 0)$. Using Table 1, we get

$$O(\overline{\boldsymbol{y}_j}, \boldsymbol{m}_j) = \frac{m_j^L}{2\overline{y}_j^L} + \frac{m_j^R}{2\overline{y}_j^R}.$$

These examples illustrate the non-quadratic objective function (10) for continuous distributions and non-logit objective functions for discrete distributions. All distributions considered are generally already implemented in statistical software, such as in the `glm()` function for the R statistical software (R Core Team, 2021).

2.3.2 Taking weights into account

Slightly adapting Theorem 3.1 of Brouste et al. (2020) allows the use of weighted MLE which can deal with other probability distributions of the exponential family. Using the proof in Appendix A.3, the MLE $\widehat{\boldsymbol{\theta}}(s) = (\hat{\theta}_L(s), \hat{\theta}_R(s))$ is obtained by changing arithmetical means $\overline{y}_j^{L/R}(s)$ to weighted means $\overline{y}_{j,w}^{L/R}(s)$ in Equation (10) according to Table 3.

From a computational point of view, the weighted sums and the weighted average can be computed as follows. Let us first compute the total weight and the total weighted sum

$$m_w = \sum_{i=1}^{n} w_i, \;\; S_w = \sum_{i=1}^{n} w_i y_i.$$

	Left node	Right node
Frequency	$m^L_{j,w}(s) = \sum_{i=1}^{n} w_i 1_{\{i\in L(j,s)\}}$	$m^R_{j,w}(s) = \sum_{i=1}^{n} w_i 1_{\{i\in R(j,s)\}}$
Average	$\overline{y}^L_{j,w}(s) = \frac{1}{m^L_{j,w}(s)} \sum_{i=1}^{n} w_i y_i 1_{\{i\in L(j,s)\}}$	$\overline{y}^R_{j,w}(s) = \frac{1}{m^R_{j,w}(s)} \sum_{i=1}^{n} w_i y_i 1_{\{i\in R(j,s)\}}$

Table 3: Notations for conditional weighted frequency and average

Then only for the left node, using Table 3, we deduce $m^L_{j,w}(s)$ and $S^L_{j,w}(s)$. Since the right-branch statistics can be deduced from m_w and $m^L_{j,w}(s)$, S_w and $S^L_{j,w}(s)$, we easily obtain the needed weighted averages as

$$\overline{y}^L_{j,w}(s) = \frac{S^L_{j,w}(s)}{m^L_{j,w}(s)}, \ \overline{y}^R_{j,w}(s) = \frac{S_w - S^L_{j,w}(s)}{m_w - m^L_{j,w}(s)}.$$

This may reduce the computation time if the selection $\{i \in L(j,s)\}$ is computer intensive. Now we present usual distributions that benefit from using a weighted MLE in order to illustrate this framework.

The binomial distribution enters the exponential family framework by considering the random variable Y/m when $Y \sim \mathcal{B}(m,p)$ for a size parameter m and a probability parameter p, see, e.g., Zeileis and Hornik (2007). Using Table 1, the distribution of Y/m belongs to the exponential family, for which $\mathbb{Y} = \{0, 1/m, \ldots, 1\}$ and $\Lambda = \mathbb{R}$. In practice, the size parameter m is known for the binomial experiment and does not have to be estimated. Furthermore, the parameter is not common to every observations, that is we model Y_i/m_i when $Y_i \sim \mathcal{B}(m_i, p_i)$. So, we can tackle this issue by considering a weighted MLE without changing a, b function yet c becomes $c(x,\phi) = 1/m_i \log\binom{m_i}{m_i x}$. Using Table 1, the objective function for the binomial distribution is the same as for the Bernoulli distribution by changing arithmetical means $\overline{y}^{L/R}_j(s)$ to weighted means $\overline{y}^{L/R}_{j,w}(s)$.

2.3.3 Other special cases for log transformed variable

Finally, we close our overview of distributions belonging to the GLM framework by considering a transformation of the response variable. Various parametrized transformations have been proposed to nonlinearly transform response variables so that the resulting distribution has a tractable distribution, typically the Gaussian distribution. In that perspective, a classical transform is the Box-Cox transform (Box and Cox, 1964). Transforming the response variable is different than transforming the expectation through a link function (possibly parametrized), see the discussion of McCullagh and Nelder (1989, Section 11.3.3).

In order to keep tractable solutions of MLE, we consider the log transformation which leads to known distributions. More precisely, we consider the transformation $t(x) = \log(d_1 x + d_2)$ and denote by $T_i = t(Y_i)$ the transformed random variables, where d_1, d_2 are known parameters, i.e., no estimation is needed. We assume that $T_1, \ldots, T_n$ are independent random variables with a distribution defined in Equation (1).

In Appendix A.4, we show that the log-likelihood (1) only differs by a new $\tilde{c}$ function

$$\tilde{c}(y,\phi) = c(y,\phi) + \log\left(\frac{d_1}{d_1 y + d_2}\right),$$

whereas a and b remain identical to the original distribution. As shown in Brouste et al. (2020), the MLE is still explicit and the closed-form expression (9) has to be updated by replacing the original variable Y_i by the transformed variable T_i. Thus, we obtain expressions given in Table 4.

	Left node	Right node
Average	$\overline{t}^L_j(s) = \frac{1}{m^L_j(s)} \sum_{i=1}^{n} t(y_i) 1_{\{i\in L(j,s)\}}$	$\overline{t}^R_j(s) = \frac{1}{m^R_j(s)} \sum_{i=1}^{n} t(y_i) 1_{\{i\in R(j,s)\}}$

Table 4: Notations for conditional average for transform $t(x)$ with frequencies given in Table 2

As for non-transformed responses, the fitted log-like-lihood is explicit, see Appendix A.4. Hence, the objective function, when searching for the best split point $s^\star$ or the best subset $s^\star$, is similar to (10)

$$\begin{aligned} O(\overline{\boldsymbol{t}_{j^\star}}(s), \boldsymbol{m}_{j^\star}(s)) = \tilde{b}\left(\bar{t}^L_{j^\star}(s)\right) m^L_{j^\star}(s)\bar{t}^L_{j^\star}(s) - b\left(\tilde{b}(\bar{t}^L_{j^\star}(s))\right) m^L_{j^\star}(s) \\ + \tilde{b}\left(\bar{t}^R_{j^\star}(s)\right) m^R_{j^\star}(s)\bar{t}^R_{j^\star}(s) - b\left(\tilde{b}(\bar{t}^R_{j^\star}(s))\right) m^R_{j^\star}(s). \end{aligned} \tag{11}$$

We present now important distributions which fall in this framework. The Pareto 1 distribution with a known threshold parameter d is obtained by considering $d_1 = 1/d$ and $d_2 = 0$. Indeed as studied in Brouste et al. (2020, Section 4), in that case, Y_i follows a Pareto 1 with shape parameter λ and threshold parameter d implies that $T_i = \log(Y_i/d)$ follows an exponential distribution $\mathcal{E}(\lambda)$, i.e., a special of the gamma distribution $\mathcal{G}(1, 1/\lambda)$. Hence, this falls in the GLM framework with a unitary dispersion $\phi = 1$ in Table 1. Using Equation (11), we obtain the following objective function

$$O(\overline{\boldsymbol{t}_{j^\star}}(s), \boldsymbol{m}_{j^\star}(s)) = -m^L_{j^\star}(s) - \bar{t}^L_{j^\star}(s)m^L_{j^\star}(s) - m^R_{j^\star}(s) - \bar{t}^R_{j^\star}(s)m^R_{j^\star}(s).$$

The shifted lognormal distribution with a known threshold parameter d is obtained by considering $d_1 = 1$ and $d_2 = -d$. Indeed as studied in Brouste et al. (2020, Section 5), in that case, Y_i follows a shifted lognormal distribution with parameters μ, σ^2 and threshold parameter d implies that $T_i = \log(Y_i - d)$ follows a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. Using Table 1 and Equation (11), we obtain the following objective function

$$O(\overline{\boldsymbol{t}_{j^\star}}(s), \boldsymbol{m}_{j^\star}(s)) = \frac{m^L_{j^\star}(s)}{2}\left(\bar{t}^L_{j^\star}(s)\right)^2 + \frac{m^R_{j^\star}(s)}{2}\left(\bar{t}^R_{j^\star}(s)\right)^2.$$

2.4 Extensions to more complex trees

In this section, we discuss two possible extensions for GLM-based trees by including more than two splits in Section 2.4.1 and several categorical explanatory variables in the GLM in Section 2.4.2.

2.4.1 Split into more than two segments

Our approach can easily deal with multiway splits at tree nodes. Generalizing the linear predictor (8), we use a generic notation for the linear predictor

$$\eta_i = \theta_{L_1} \times 1_{\{i \in L_1(j,s)\}} + \cdots + \theta_{L_m} \times 1_{\{i \in L_m(j,s)\}}, \tag{12}$$

where $L_k(j, s)$ is the k-th leaf subset resulting from the split. For numeric partitioning variables, $L_1 \cup \cdots \cup L_m$ is a partition of the interval $[\min_i z_{i,j}, \max_i z_{i,j}]$, while for categorical partitioning variables, it is a partition of the modalities set $\{v_{j,1}, \ldots, v_{j,l_j}\}$.

Again, from Theorem 3.1 of Brouste et al. (2020), the MLE $\widehat{\boldsymbol{\theta}}(s)$ depends only on the link function g and is given by

$$\hat{\theta}_{L_k}(s) = g\left(\overline{y}^{L_k}_{j^\star}(s)\right), \; m^{L_k}_{j^\star}(s) = \sum_{i=1}^n 1_{\{i \in L_k(j^\star,s)\}},$$

$$\overline{y}^{L_k}_{j^\star}(s) = \frac{1}{m^{L_k}_{j^\star}(s)} \sum_{i=1}^n y_i 1_{\{i \in L_k(j^\star,s)\}}.$$

The fitted log-likelihood is also explicit using Corollary 3.1 of Brouste et al. (2020) and the objective function (10) is generalized to

$$O(\overline{\boldsymbol{y}_{j^\star}}(s), \boldsymbol{m}_{j^\star}(s)) = \sum_{k=1}^m \tilde{b}\left(\overline{y}^{L_k}_{j^\star}(s)\right) m^{L_k}_{j^\star}(s)\overline{y}^{L_k}_{j^\star}(s) - \sum_{k=1}^m b\left(\tilde{b}(\overline{y}^{L_k}_{j^\star}(s))\right) m^{L_k}_{j^\star}(s). \tag{13}$$

This objective function is completely tractable and deals with both categorical and numerical partitioning variables. This differs from other algorithms for multiway splits. Indeed, CHAID by Kass (1980) and CRUISE by Kim and Loh (2001) build up contingency tables to compute chi-square statistics in the

search of the best partition not restricted to two children nodes. FACT by Loh and Vanichsetakul (1988) is based on the linear discriminant analysis which allows multiple splits at the same node. Note that categorical partitioning variables in CRUISE and FACT are converted to numerical variables via the largest discriminant coordinate. Functional trees (FT) by Gama (2004) provide a general algorithm but not an explicit objective function to be maximized at each node. LMT by Landwehr et al. (2005) deals only with Bernoulli or multinomial responses and propose an iterative algorithm (LogitBoost) to fit logistic regression models as well as heuristics to speed up their algorithm.

2.4.2 Explicit MLE solutions with categorical explanatory variables

Explanatory variables can be included in the GLM as it is permitted by MOB. For simplicity, we modify Equation (8) by introducing a single explanatory categorical variable[3]. Similarly to Brouste et al. (2020, Section 3.1), we denote by $x_i^{(1)}$ the explanatory categorical variable which takes values in a set $\{v_1, \ldots, v_d\}$ of size $d > 2$. Considering m splits, the linear predictor can be simplified

$$\eta_i = \sum_{k=1}^{m} \sum_{l=1}^{d} 1_{\{i \in L_k(j,s)\}} 1_{\left\{x_i^{(1)} = v_l\right\}} \theta_{L_k}^l = \sum_{k=1}^{m} \sum_{l=1}^{d} x_i^{(k,l)} \theta_{L_k}^l, \tag{14}$$

where $\theta_{L_k}^l$ for different k and l are unknown parameters and $x_i^{(k,l)} = 1_{\{i \in L_k(j,s), x_i^{(1)} = v_l\}}$ is a new dummy.

Using Example 3.4 of Brouste et al. (2020), the exact MLE of $\boldsymbol{\theta}$ is for a given split point s

$$\hat{\theta}_{L_k}^l(s) = g(\overline{y}_{j^\star}^{k,l}(s)), k = 1, \ldots, m, l = 1, \ldots, d,$$

where $\overline{y}_{j^\star}^{k,l}$ is the empirical average over node k and modality v_l given the partitioning variable $j^\star$. Again the fitted log-likelihood can be derived using Corollary 3.3 of Brouste et al. (2020): this consists in replacing averages $\overline{y}_{j^\star}^{L_k}(s)$ by $\overline{y}_{j^\star}^{k,l}(s)$ as well as node sides $m_{j^\star}^{L_k}(s)$ by $m_{j^\star}^{k,l}(s)$ in (13) .

With a higher number of explanatory categorical variables, we may define dummy variables based on interaction terms as in (14). However, an explicit log-likelihood is available only when the assumptions of Theorem 3.2 of Brouste et al. (2020) are satisfied. More precisely, for two categorical explanatory variables with modality numbers $d_2 > 1$, $d_3 > 1$, the MLE $\hat{\theta}$ is given by solving a system zeroing the score where we need to impose that $q = 1 + d_2 + d_3$ linear constraints on the parameter $\boldsymbol{\theta}$ ($R\boldsymbol{\theta} = \mathbf{0}$) as well as

$$\det(R_1 - Q_1 R_2) \neq 0,$$

where $Q_1 = (\mathbf{1}_{d_2 d_3}, \mathbf{1}_{d_3} \otimes I_{d_2}, I_{d_3} \otimes \mathbf{1}_{d_2})$ is a $d_2 d_3 \times (1 + d_1 + d_2 + d_2 d_3)$ real matrix with $d_2 d_3$ the cardinal of the set of combined modalities between the two categorical explanatory variables (excluding unobserved combinations), $R = (R_1, R_2)$ is a $q \times (1 + d_1 + d_2 + d_2 d_3)$ real matrix of linear contrasts whose rank is equal to q, $\otimes$ is the Kronecker product, I_{d_2} and I_{d_3} are identity matrices and $\mathbf{1}_{d_2}, \mathbf{1}_{d_2}$ and $\mathbf{1}_{d_2 d_3}$ are ones-vectors. In the case of no intercept and no single-effect, this simplifies and the linear predictor (14) is generalized to two explanatory variables as

$$\eta_i = \sum_{k=1}^{m} \sum_{l_2=1}^{d_2} \sum_{l_3=1}^{d_3} x_i^{(k,l_2,l_3)} \theta_{L_k}^{l_2,l_3},$$

where $\theta_{L_k}^{l_2,l_3}$ are unknown parameters and

$$x_i^{(k,l_2,l_3)} = 1_{\{i \in L_k(j,s), x_i^{(1)} = v_{l_2}, x_i^{(2)} = v_{l_3}\}},$$

is a new dummy variable. In that case, its MLE is obtained as the g-transformed average response over L_k given $x_i^{(1)} = v_{l_2}, x_i^{(2)} = v_{l_3}$, i.e. for a given split point s, $k = 1, \ldots, m, l_1 = 1, \ldots, d_1$ and $l_2 = 1, \ldots, d_2$

$$\hat{\theta}_{L_k}^{l_2,l_3}(s) = g(\overline{y}_{j^\star}^{k,l_2,l_3}(s)).$$

[3] We do not consider an intercept as the fitted log-likelihood is identical with or without intercept, see Brouste et al. (2020, Corollary 3.1).

To our knowledge, this approach can not be directly extended to one or more continuous explanatory variables, since explicit MLE is only available for categorical explanatory variables. Different ways are possible for bypassing this issue. In some situations, continuous explanatory variables are never used directly but splitted into levels, typically based on a regular grid or on quantile values. In experimental situations also, continuous explanatory variables are sometimes recorded with a finite precision such as for concentration, temperature, volume, so that a categorization of the variable is practicable. For instance, this is the case for insurance pricing where the policyholder age is discretized, e.g., Denuit et al. (2019). In ecotoxicology, Szöcs and Schäfer (2015) use discrete values in mg/L of treatment to explain mayfly larvae counts. Alternatively, if modelers want to benefit from explicit MLE for GLM when building the tree, continuous variables could be used as partitioning variables. Finally, the original glmtree model (without explicit solutions) can of course be used in situations where continuous explanatory variables should remain numeric so that a single coefficient is fitted in order to obtain monotonicity of the response with respect to that variable.

3 Computational gain of closed-form solutions for GLM trees

In this section, we want to compare the performances, both in terms of accuracy and computation time, of model-based recursive partitioning methods against a CART model on simulated datasets. Following the literature (Zeileis et al., 2008), the superiority of GLM-based trees is expected, but our approach should significantly reduce the runtime.

3.1 Dataset simulation

We consider an approach to build up various test datasets used in Wood (2011) for benchmarking generalized additive models. Precisely for m explanatory variables, we first generate independent and uniformly random variables $(x_{i,j})_{i,j}$. Then, we simulate continuous independent variables Y_i, $i = 1, \dots, n$ with the mean $\mu_i = g^{-1}(\eta_i)$ for a linear predictor η_i defined as

$$\eta_i = 1 + \sum_{j=1}^{m} f_{j-1(\text{mod } 15)}(x_{i,j}),$$

where the first five f_j are Simon Wood's smooth functions and the last ten f_j are similar nonlinear test functions defined in Appendix B.1. Distributions considered are given in Table 5. For the considered distributions, a dispersion ϕ is needed to fully characterize the distribution, see last column in Table 5 for ϕ used when generating datasets and Table 1 for the link between the shape parameter and ϕ. Various sample sizes are considered when simulating ranging from $n = 100$ to $n = 50000$ as well as different explanatory variable number $m = 10$ or 20. The naming convention for simulated datasets is given in Table 6, for instance `contIG2` refers to a simulated dataset with $m = 20$ continuous explanatory variables with inverse Gaussian responses.

Distribution	μ_i	ϕ
Gaussian $\mathcal{N}(\mu_i, \sigma^2)$	$\mu_i = \eta_i$	0.25
gamma $\mathcal{G}(\nu, \mu_i)$	$\mu_i = e^{\eta_i/5}$	0.25
inverse Gaussian $\mathcal{IG}(\mu_i, \sigma^2)$	$\mu_i = e^{\eta_i/5}$	0.1

Table 5: Mean and dispersion parameters μ, ϕ used in simulations

`variable`	`distribution`	`number`
`cont` for continuous expl. variables	`IG` for inverse Gaussian	1 for $m = 10$ covariates
`categ` for categorical expl. variables	`G` for gamma	2 for $m = 20$ covariates
	`N` for normal	

Table 6: Naming convention for datasets `<variable><distribution><number>`

3.2 Implementation details

Numerical illustrations of this paper have been carried out in the R statistical software. Core functions, such as `glm`, are implemented in the **stats** package, but some CRAN packages are also used. We use `ctree` from package **partykit** by Hothorn and Zeileis (2015) to fit recursive partitioning based conditional inference trees (*CTREE*), and `rpart` from package **rpart** by Therneau and Atkinson (2019) for the CART algorithm. New functions have been written to compare GLM-based trees: in the following, *GLM tree* refers to outputs of `glmtree`, a special case of `mob`, in **partykit**, and *explicit GLM tree* refers to the new estimation procedure for GLM trees proposed in Section 2. In addition, the function `lmtree` from package **partykit** refers to linear model tree, which is equivalent to GLM trees with a Gaussian distribution. Recall that a closed-form solution exists for this last method. Variables $(x_{i,j})_{i,j}$ can play the role of both explanatory and partitioning variables in the GLM-tree algorithm. All theses variables are considered as partitioning variables in Section 3.3 and 3.4, i.e., GLMs are fitted with an intercept. GLMs with one explanatory variable are also considered in Section 3.4. They are estimated with our *Explicit GLM Tree* model if the explanatory variable is categorical, and with *GLM Tree* if the explanatory variable is continuous. All analyses are done with the default parameters with the following exceptions. All trees are constrained to have a minimum number of 7 observations per terminal node, 20 observations per internal node, and a maximum tree depth of 9 for all models. Furthermore, we also considered three options[4] of `ctree` fit by modifying the way the distribution of the test statistic is computed: *Teststatistic* refers to the raw statistic for computing the p-values, *Bonferroni* and *Univariate* correspond respectively to adjusted and unadjusted p-values from the asymptotic distribution. It is worth noting that the tuning of parameters has not been optimized. A wise choice of these parameters could improve the prediction quality of the different models.

3.3 Runtime comparison between *GLM tree* and *explicit GLM tree*

In this section, we especially focus on the gain in runtime between the original GLM tree method and our new approach. To examine the superiority of our approach in terms of computation speed, we perform a comparison simulation as described above and generate several samples of different sizes. Here, performance are assessed based on the computation time only since the predictions are identical.

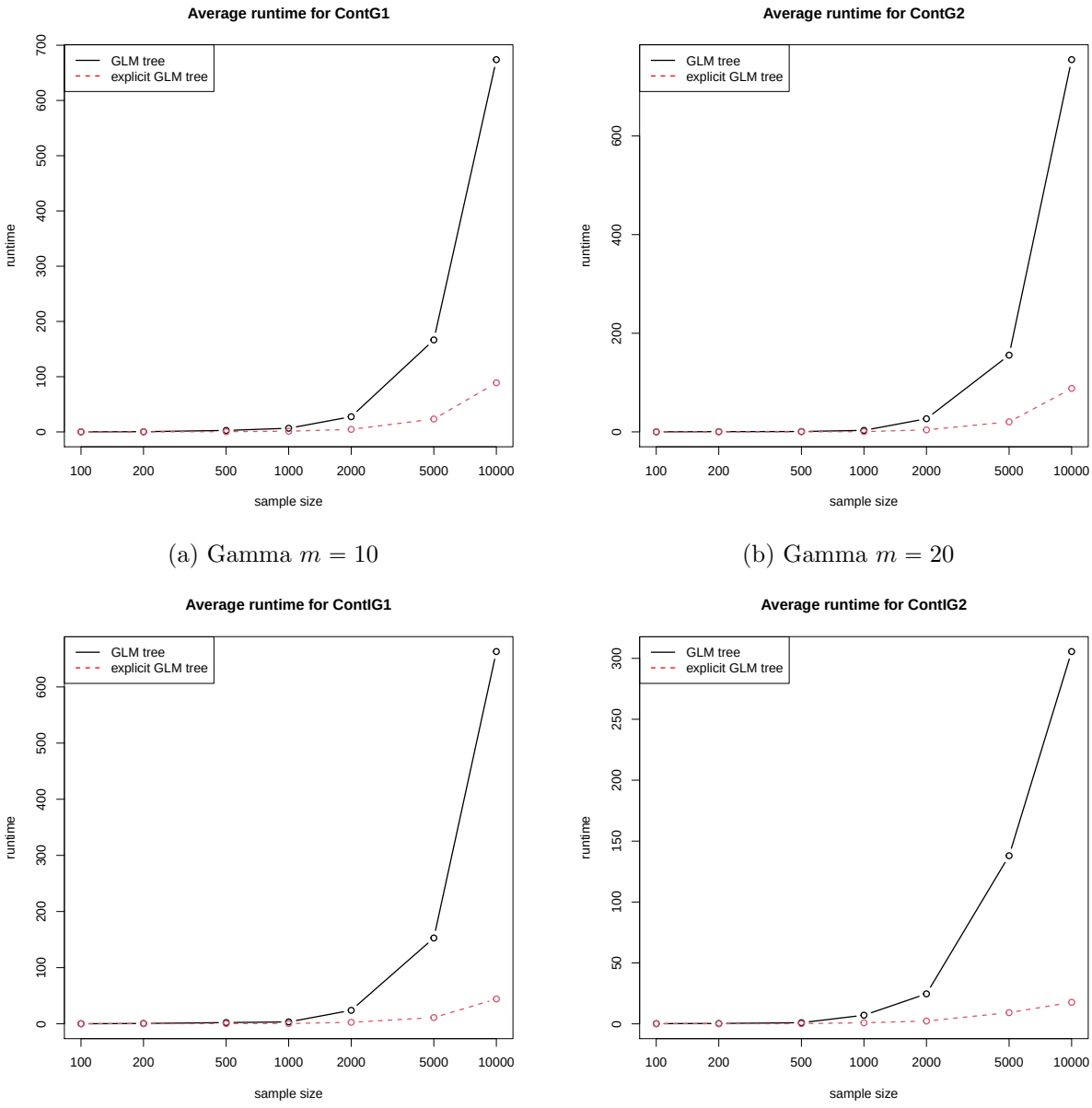
Figure 1 displays average computation times (over 10 runs) as a function of sample size from $n = 10^2$ to 10^4 for ten continuous explanatory variables. Sub-figures 1a and 1b correspond to the Gamma distribution while Sub-figures 1c and 1d are for the inverse Gaussian distribution. The results presented show clearly the important gain in computation time for all sub-figures of Figure 1 when the size of the simulated sample increases. For sample size larger $n > 10^3$, *explicit GLM tree* is particularly fast. Note also that comparisons cannot be done on samples of larger size because `glmtree`, which uses `glm`, does not converge when GLMs are fitted during the split point search. This advantage holds irrespectively of the chosen distribution or the number of explanatory variables.

Time ratios for other distributions and link functions are displayed in Figure 6 in Appendix B.2. Note that for some distributions, such as normal or gamma, *GLM tree* does not converge for a non-canonical link so that a comparison with *explicit GLM tree* is unfortunately not possible. Non-convergence issues for *GLM tree* is due to some combinations of split points and explanatory variables where `glm` is badly initialized and does not reach the convergence. Time ratios for the mean, the median and the standard deviation runtime have similar patterns leading to the same conclusion.

3.4 Comparison with benchmark models

Since CART models are often preferred to unbiased tree methods while accuracy may be worse, our aim is to show how the trade-off between computation times and predictive accuracy evolves with the introduction of our approach. We compare the performance in accuracy, complexity and computation time of our version of GLM trees with an explicit solution (*Explicit GLM Tree*) with a intercept-node only; GLM trees with a one explanatory variable (*GLM Tree reg*) which admit an explicit solution when

[4]The p-values can be computed by Monte-Carlo and then adjusted with Bonferroni. This fourth option has also be examined, but the results is not reported as the goodness-of-fits of results is comparable to the other `ctree` options, but the computation times is much longer.



(a) Gamma $m = 10$

(b) Gamma $m = 20$

(c) Inverse Gaussian $m = 10$

(d) Inverse Gaussian $m = 20$

Figure 1: Computation time (sec) of *GLM tree* and *explicit GLM tree*

this variable is categorical; the original approach based on IWLS (*GLM Tree*), *CTREE* with different test specifications and *CART*. These performance indicators are assessed by performing a bootstrap cross-validation approach based on 100 bootstrap replications with replacement for each dataset. The sample size is fixed to $n = 1,000$. Then, different models are trained on each bootstrap sample and validated based on the leave-out cross-validation estimates. The accuracy of each model is calculated through the root mean squared error (MSE) since the response is continuous. Complexity of trees is assessed as the number of its terminal nodes. Finally, we assess the computation times in seconds for each of these methods.

Figure 2 displays the distribution of the predictive RMSE of the considered methods over 4 simulated data-sets introduced in Table 6, namely `CategG1`, `CategIG1`, `ContG1` and `ContIG1`. For almost all datasets, the *explicit GLM Tree* and *GLM Tree*[5] with inverse Gaussian or gamma distributions produces the smallest median RMSE compared to the other specifications. This result can be expected since inverse Gaussian or gamma distributions are used for generating the response variable [6]. The performance of

[5]Since the results of *explicit GLM Tree* and *GLM Tree* are equal, only those related to *explicit GLM Tree* are depicted.

[6]We also model a post-pruned version of the CART model using the one standard error criterion, which produce more

GLM Tree reg with one explanatory variable are fairly similar for `CategG1` and `CategIG1`, except for the Gamma distribution, and for `ContIG1`. We observe however than the *explicit GLM Tree* performs better for `ContG1`. We also note that the *explicit GLM Tree* with Gaussian distribution outperforms *CART* and the CTREE specifications for for all samples, even if the difference is visually small with the latter for `CategIG1` and `ContG1`. Differences are observed between the results of *LM Tree* and those of *Explicit GLM Tree* with Gaussian distribution. These discrepancies, which appear in a material way only for the `ContIG1` dataset, are explained by the splitting criterion which is different between a linear model and a generalized linear model. Indeed, the first one is based on the residual sum of squares while the second one is based on the log-likelihood of the model. Furthermore, the interquartile range of the predictive RMSE does not show that *Explicit GLM tree* models provide more reliable predictions than other algorithms.

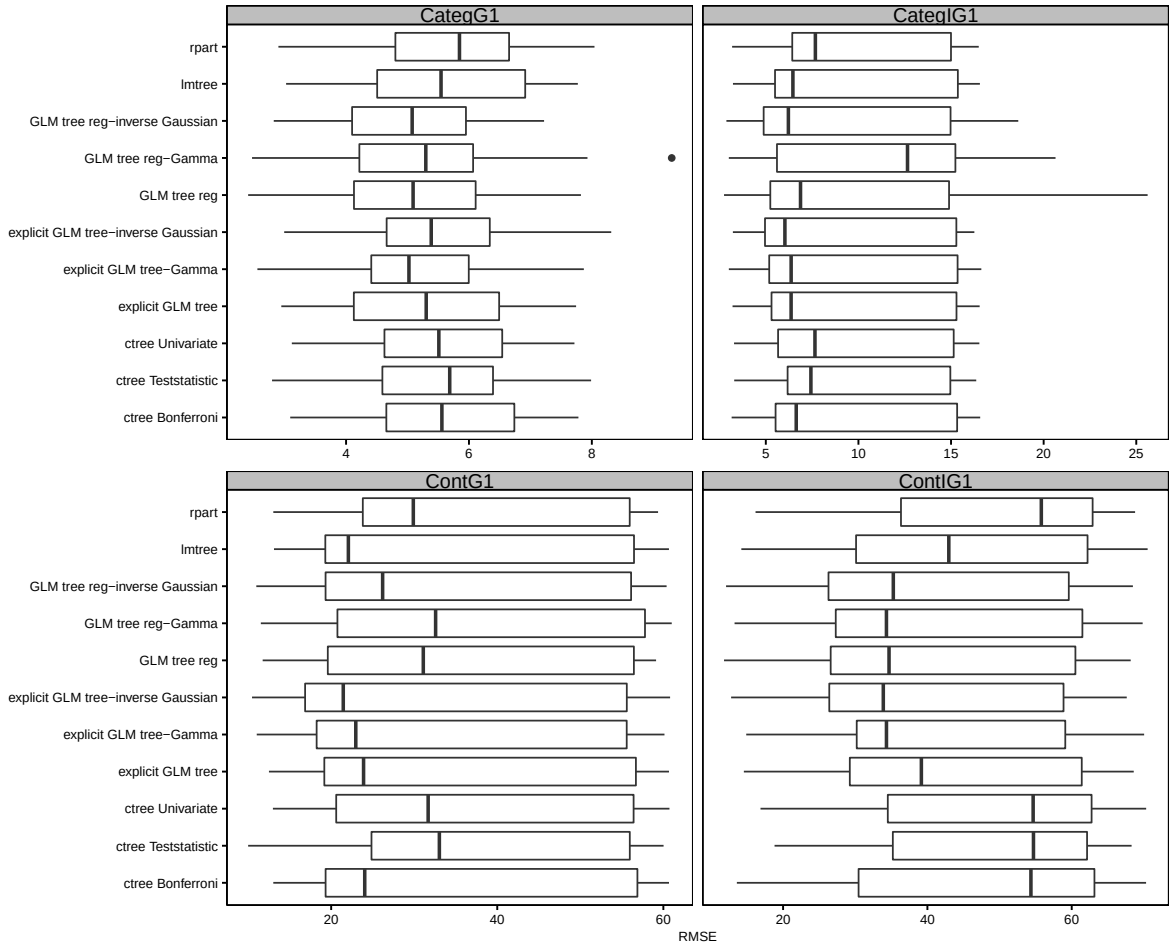


Figure 2: Predictive RMSE with 100 bootstrap replications, `CategG1`: gamma with categorical var.; `CategIG1`: inv. Gaussian with categorical var.; `ContG1`: gamma with continuous var.; `ContIG1`: inv. Gaussian with continuous var.

Table 7 focuses on the average over the 100 bootstrap samples of the predictive complexity. Despite its lesser performance, we clearly note that *CART* produces less complex trees compared to other models. As a result, this model retains an important interest compared to its competitors to the extent that *CART* provides more interpretable and simpler to explain results. The complexity of all *Explicit GLM Tree* and *LM tree* are close for each of them, but much higher to that of *CART*. For instance, on the `CategG1` dataset, the average complexity is around 3.3 times higher than the number of terminal nodes of *CART*. Hence, the choice of these tree models to be used can be assessed in terms of the trade-off between performance and complexity. The complexity of a *Explicit GLM Tree* and a *GLM Tree reg*

accurate results for CART model on the `CategIG1` and the `ContIG1` datasets, but *explicit GLM Tree* remains better in terms of RMSE.

is similar with categorical variables, while a *GLM Tree reg* is clearly less complex in presence of one continuous explanatory variable and continuous partitioning variables. This comes from the fact that the trees of *GLM Tree reg* are grown with one less partitioning variable than *Explicit GLM Tree*, which clearly reduce the number of possible nodes for trees built on continuous partitioning variables. The complexity results of CTREE models depends on the chosen options for the variable selection test and can be higher than the complexity of *Explicit GLM Tree*. For all datasets, the Bonferroni option is the most parsimonious without a loss of performance, see Figure 2. Regarding the complexity, *Explicit GLM Trees* are less complex to this specification on datasets with categorical partitioning variables (`CategG1` and `CategIG1`) and more complex for datasets with partitioning continuous variables.

Method	`CategG1`	`CategIG1`	`ContG1`	`ContIG1`
ctree Bonferroni	51.770 (4.156)	64.570 (4.841)	7.880 (3.006)	18.690 (3.826)
ctree Teststatistic	77.430 (2.417)	77.520 (2.834)	53.000 (8.299)	68.560 (5.907)
ctree Univariate	64.880 (3.264)	71.910 (3.232)	17.830 (5.520)	36.950 (6.559)
GLM tree reg	30.990 (1.982)	32.220 (1.643)	5.520 (3.301)	13.660 (5.769)
GLM tree reg-Gamma	30.022 (2.022)	32.225 (1.814)	5.340 (2.886)	12.420 (6.054)
GLM tree reg-inverse Gaussian	30.391 (1.827)	32.261 (1.725)	5.140 (3.291)	13.210 (5.186)
explicit GLM tree	31.640 (1.738)	31.090 (2.327)	10.950 (2.858)	25.800 (3.162)
explicit GLM tree-Gamma	31.450 (1.714)	31.170 (2.070)	10.220 (2.939)	25.030 (3.395)
explicit GLM tree-inverse Gaussian	31.420 (1.742)	30.800 (2.429)	12.560 (3.016)	22.990 (3.611)
lmtree	32.010 (1.888)	32.130 (2.377)	11.370 (3.139)	24.130 (3.620)
rpart	9.530 (1.795)	6.880 (3.291)	5.040 (2.558)	5.360 (1.494)

Table 7: Mean predictive complexity over 100 bootstrap replications with standard deviations in parentheses. `CategG1`: gamma with categorical var.; `CategIG1`: inv. Gaussian with categorical var.; `ContG1`: gamma with continuous var.; `ContIG1`: inv. Gaussian with continuous var.

The mean runtime is summarized in Table 8. The results of *Explicit GLM Tree* and *GLM Tree* are presented separately since the computation times obtained are different. In a similar way to Section 3.3, it can be observed that *Explicit GLM Tree* largely outperforms *GLM Trees*. Concerning the *CTREE*, the performance gap with *GLM Trees* differs depending on whether the partitioning variables are continuous or categorical. For the `CategG1` dataset for example, the models with a Gamma distribution and an inverse Gaussian distribution have an average computation time of respectively 0.658 seconds and 0.612 seconds against 0.541 seconds for a *CTREE* with the Bonferroni option. The improvement is thus substantial compared to the use of `glmtree` for which the computation time is 2.45 times and 2.65 times higher on average. A comparable result is observed with the `CategIG1` dataset. Furthermore, it is also interesting to compare the performance of *Explicit GLM Trees* and *GLM Trees reg* for categorical and continuous variables. With categorical variables, *GLM Trees reg* admit an explicit formula and clearly outperforms *Explicit GLM Tree*. For continuous data, although the *Explicit GLM Tree* algorithm relies on an explicit formula, it does not outperform *GLM Tree reg* due to its higher complexity.

For continuous partitioning variables, the superiority of *CTREE* over *Explicit GLM Tree* is more marked. For the `ContG1` dataset for example, the models with a Gamma distribution and an inverse Gaussian distribution have an average computation time of 3.826 seconds and 2.887 seconds respectively against 0.089 seconds for a *CTREE* with the Bonferroni option. In this situation, the gaps in terms of computation time between the two types of algorithms is amplified since the number of split points is greater than for categorical partitioning variables. The gain compared to `glmtree` due to the use of a closed–form formula remains however very interesting. In addition, we can observe that the computation time of *CART* is much lower than those of the other algorithms. The **rpart** package is indeed known for its speed of execution as it relies on C code, whereas the **partykit** package is entirely developed in the R language, which largely explains the important differences of calculation times.

3.5 Performance on `BostonHousing` and `Hitters` datasets

We now assess the performance of *Explicit GLM tree*, *CTREE* and *CART* on two public benchmark datasets: `BostonHousing` and `Hitters` from R packages **mlbench** (Leisch and Dimitriadou, 2021) and **ISLR** (James et al., 2017). On these datasets, most of variables are continuous, including the response

Method	CategG1	CategIG1	ContG1	ContIG1
ctree Bonferroni	0.541 (0.247)	0.418 (0.073)	0.089 (0.043)	0.162 (0.041)
ctree Teststatistic	0.682 (0.326)	0.471 (0.079)	0.365 (0.138)	0.415 (0.084)
ctree Univariate	0.622 (0.312)	0.453 (0.091)	0.166 (0.081)	0.262 (0.068)
GLM tree reg	0.372 (0.043)	0.484 (0.068)	2.310 (1.215)	3.095 (1.071)
GLM tree reg-Gamma	0.484 (0.067)	0.517 (0.070)	2.214 (1.076)	2.980 (1.350)
GLM tree reg-inverse Gaussian	0.436 (0.069)	0.471 (0.064)	2.141 (1.455)	3.590 (1.766)
explicit GLM tree	0.922 (0.452)	0.687 (0.170)	2.943 (1.711)	3.047 (0.461)
explicit GLM tree-Gamma	0.658 (0.099)	0.674 (0.104)	3.826 (0.651)	4.494 (0.585)
explicit GLM tree-inverse Gaussian	0.612 (0.102)	0.632 (0.134)	2.887 (0.659)	2.761 (0.338)
GLM tree	1.121 (0.525)	0.827 (0.163)	5.243 (1.857)	5.708 (0.828)
GLM tree-Gamma	1.325 (0.204)	1.326 (0.197)	17.887 (2.517)	18.601 (1.887)
GLM tree-inverse Gaussian	1.433 (0.227)	1.432 (0.274)	16.165 (4.342)	21.721 (1.693)
lmtree	0.713 (0.357)	0.483 (0.117)	1.651 (1.017)	1.493 (0.308)
rpart	0.022 (0.009)	0.015 (0.007)	0.022 (0.015)	0.018 (0.004)

Table 8: Mean predictive runtime of different methods over 100 bootstrap replications with standard deviations in parentheses. `CategG1`: gamma with categorical variables; `CategIG1`: inverse Gaussian with categorical variables; `ContG1`: gamma with continuous variables; `ContIG1`: inverse Gaussian with continuous variables.

variable. Hence, we consider three distributions for *GLM trees* (Gaussian, inverse Gaussian and gamma) with 100 bootstrap replications as in Section 3.2.

Figure 3 displays the distribution of the predictive RMSE. For `BostonHousing`, *CTREEs* are the best, yet other algorithms performance are very comparable. Since the response variable for this dataset is the median value of owner-occupied homes in USD 1000's. Consequently, inverse Gaussian and gamma distributions do not seem well suited, which explains why *Explicit GLM trees* with these distributions do not clearly outperform the Gaussian model. For `Hitters`, *LM Tree* is the best yet all models perform similarly. Since the shape of the density of the variable distribution is fairly close to that of a gamma distribution, the specification with this law is the best for *Explicit GLM trees*.

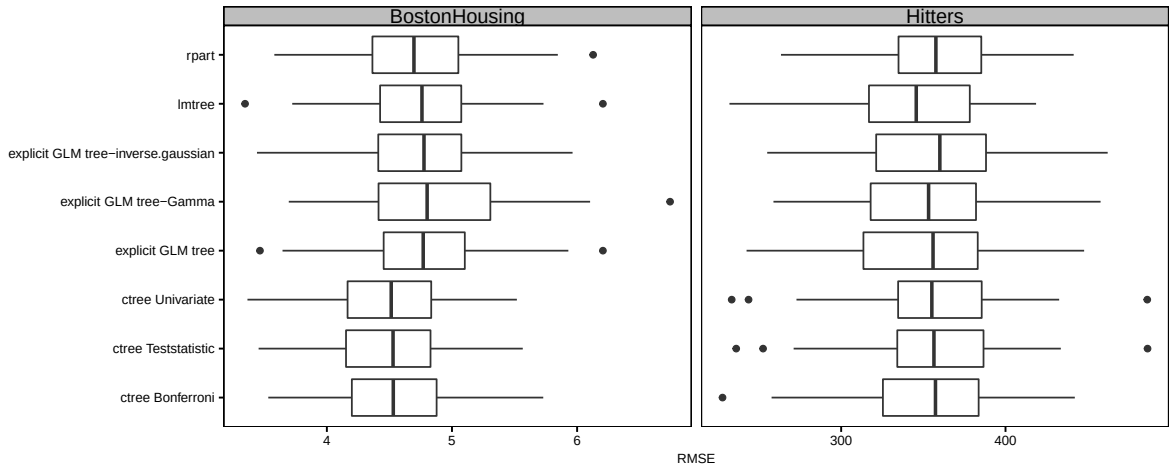


Figure 3: Predictive RMSE with 100 bootstrap replications for `BostonHousing` and `Hitters`

Table 9 displays both the average and the median over the 100 bootstrap samples of the predictive complexity and the runtime. The most complex trees are observed for *CTREE* on both datasets. Again, we observe the Bonferroni option produces the most parsimonious *CTREEs*. The least complex trees are produced by *CARTs* for `BostonHousing` and *Explicit GLM Trees* as well as *LM trees* for `Hitters`. Regarding mean and median runtimes, *Explicit GLM Trees* clearly outperform *GLM Trees* for both datasets. Since most of partitioning variables are continuous, all *CTREEs* are more efficient than *Explicit GLM Trees*, not due to differences in complexity but rather the way that the *Explicit GLM Tree* algorithm is implemented in R. Finally, we note that the *CART* algorithm remains the fastest method

due to its very efficient implementation compared to the other models.

			Complexity		Computation time	
data	method	family	mean	median	mean	median
BostonHousing	ctree Bonferroni	Gaussian	21.90	22.0	0.359	0.363
	ctree Teststatistic	Gaussian	40.91	41.0	0.523	0.525
	ctree Univariate	Gaussian	36.21	36.5	0.493	0.494
	explicit GLM tree	Gaussian	14.15	14.0	1.330	1.343
	GLM tree	Gaussian	14.15	14.0	2.171	2.216
	explicit GLM tree	Gamma	13.82	14.0	1.752	1.742
	GLM tree	Gamma	13.82	14.0	4.678	4.684
	explicit GLM tree	inverse Gaussian	14.09	14.0	1.462	1.464
	GLM tree	inverse Gaussian	14.09	14.0	4.868	4.875
	lmtree	Gaussian	13.70	14.0	0.662	0.676
	rpart	Gaussian	9.01	9.0	0.036	0.038
Hitters	ctree Bonferroni	Gaussian	9.35	9.0	0.209	0.211
	ctree Teststatistic	Gaussian	21.77	22.0	0.333	0.333
	ctree Univariate	Gaussian	18.43	19.0	0.301	0.311
	explicit GLM tree	Gaussian	7.08	7.0	0.475	0.484
	GLM tree	Gaussian	7.08	7.0	0.701	0.720
	explicit GLM tree	Gamma	6.50	6.5	0.604	0.597
	GLM tree	Gamma	6.50	6.5	1.430	1.435
	explicit GLM tree	inverse Gaussian	6.56	6.0	0.550	0.545
	GLM tree	inverse Gaussian	6.56	6.0	1.514	1.521
	lmtree	Gaussian	6.06	6.0	0.265	0.254
	rpart	Gaussian	9.47	9.0	0.030	0.030

Table 9: Complexity and runtime mean and median for `BostonHousing` and `Hitters`

4 Random forest based on GLM Trees

In this section, we assess the benefits of our approach based on a closed-form formula by implementing a random forest type approach for GLM tree model, called *GLM forest* hereafter. This analysis is conducted on simulated datasets, where we compare the performance of our approach against two classical random forest competitors: the function `cforest` from package **partykit** to fit random forests based on *CTREE*, as well as the function `randomForest` from the R package **randomForest** (Liaw and Wiener, 2002).

4.1 Datasets and implementation details

We use the following datasets `ContG2` and `ContIG2` defined in Section 3.1, that is, datasets with $m = 20$ continuous explanatory variables for gamma or inverse Gaussian responses and $n = 1000$ observations, see Table 6.

We consider three versions of *GLM forest* by choosing Gaussian, gamma and inverse Gaussian distributions (with canonical link) as for *GLM tree* in Section 3. Regarding *cforest*, we also consider three versions depending on the way the distribution of the test statistic is computed: *Teststatistic* refers to the raw statistic, *Bonferroni* and *Univariate* correspond respectively to adjusted and unadjusted p-values as in the previous section. Finally, the `randomForest` function from package of the same name is used with the default arguments, except for those listed below.

In order to make a fair and reliable comparison, we control forest-type algorithms using the following arguments:

- the number of trees, called `ntree`,
- the number of input variables randomly sampled as splitting candidates at each node, called `mtry`,
- the maximum depth of the tree, called `maxdepth`. An infinite value means that no restrictions are applied to tree sizes.

For `randomForest`, the terminal node number is capped by 2^{maxdepth} since there is no argument for the maximum depth. We consider two metrics for this benchmark: the usual root mean squared error (RMSE) as well as the mean absolute error (MAE) which proves to be a discriminant metric complementary to the RMSE. These performance indicators are assessed by performing a bootstrap cross-validation approach based on 100 bootstrap replications with replacement for each dataset as in Section 3.4.

4.2 Benchmark accuracy results

In Figure 4, we display RMSE and MAE for seven algorithms considered in this paper as a function of the maximum tree depth with `ntree=500` and `mtry=5`. Regarding RMSE, all algorithms have similar performance for both datasets. The three *cforests* are very close to one another leading to the conclusion that the test statistics has no importance on RMSE yet the computation times are different as we will see. Depending on the type of distribution used, *GLM forests* perform well for small trees on `ContIG2` in Subfigure 4b and for large trees on `ContG2` in Subfigure 4a. On these datasets, the *randomForest* algorithm proves to be less competitive with no situation where this algorithm is the best, even for unconstrained trees (infinite `maxdepth`).

Regarding MAE, we observe a large difference between *randomForests* and the others algorithms. Using this another metric leads to a viewable difference for the three *cforests* where *cforest Test Stat.* appears as the worst of all. In Subfigure 4d, the best algorithm is always a *GLM forest* algorithm, whereas for Subfigure 4c *cforest Bonferroni* is the best for medium-size trees. Overall, *GLM forests Gaussian* perform particularly well, while the datasets are generated with non-Gaussian distributions.

This analysis was also performed on datasets with less simulated explanatory variables `ContG1` and `ContIG1` with similar conclusions. We also test our seven algorithms on a wide range of tree numbers (`ntree`) and partitioning variable numbers (`mtry`) for which similar patterns of RMSE and MAE are observed.

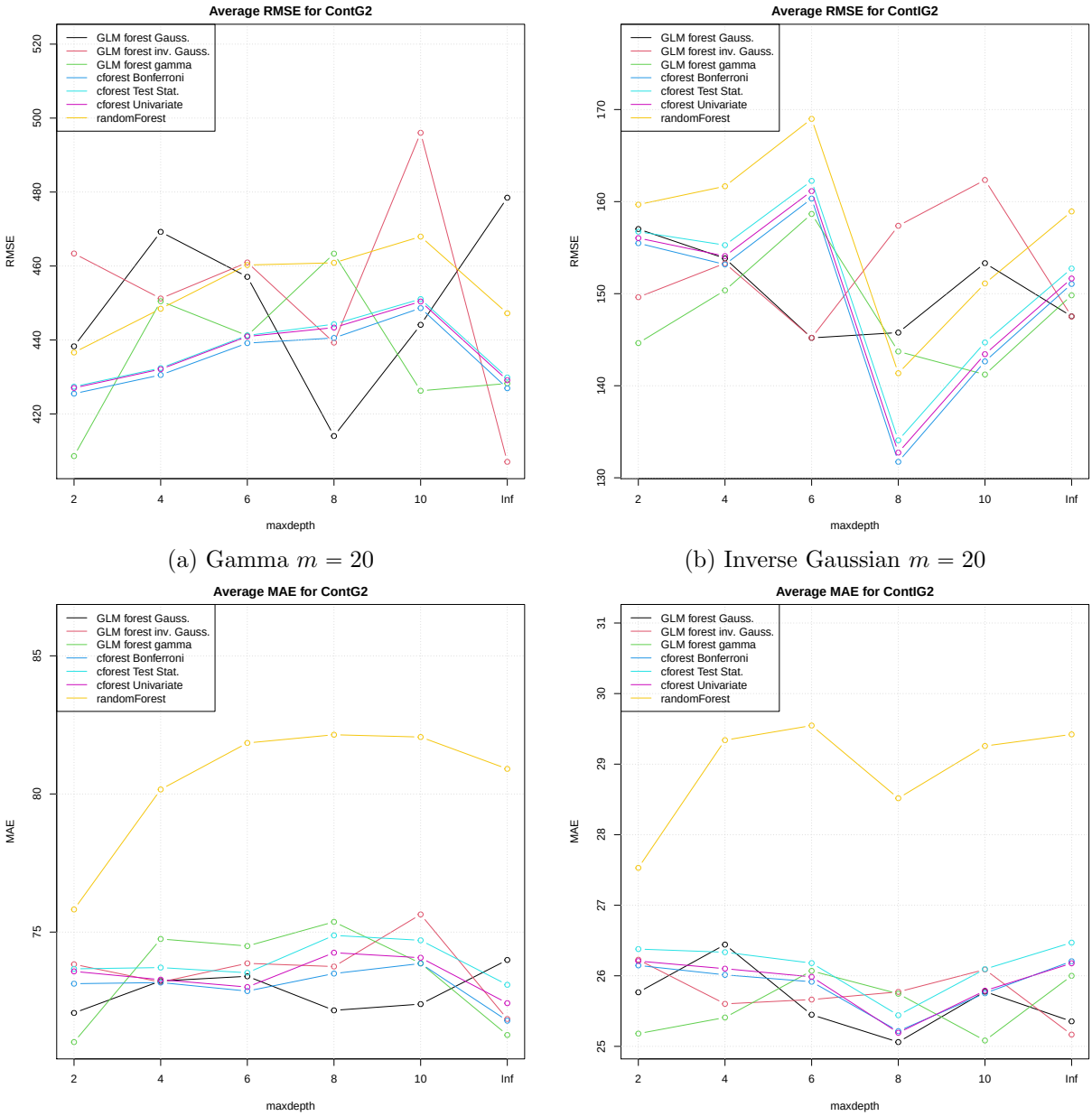
4.3 Runtime and complexity analysis

The analysis of runtime and complexity also gives another point of view of our benchmark. The complexity of a forest for a given run is the sum of terminal node number for each tree. The maximum complexity for a given maxdepth is thus $100 \times 2^{\text{maxdepth}}$. In Table 10, we display the mean and the median of complexity as well as computation times (seconds) for `maxdepth=8`.

By far, the *randomForest* algorithm proposes the most complex forests with a complexity ten times higher than that of the *cforest Teststatistic* algorithm, which is in turn ten times higher than other complexities. In conjunction with Figure 4, at `maxdepth=8`, we can conclude that most complex algorithms (such as `randomForest` and `cforest Teststatistic`) do not produce necessarily the best model in terms of RMSE, which is an interesting feature of *cforests* and *GLM forests*.

When `mtry` and `ntree` parameters are fixed, we empirically observe that the depth of trees can affect the accuracy of models, but there is no monotonous relation between the depth of trees and the predicted RMSE. Furthermore, modifying the p-value for parameter stability test of *cforests* and *GLM forests* (when a node is split) does not greatly improve the accuracy of *GLM forests*. On these datasets, forests with smaller trees (such as *GLMforests* and *cforests Bonferroni*) produce better trees, both in terms of performance and complexity.

In terms of runtime, there are three groups: *random-Forest* and *cforest Bonferroni* are by far the fastest; *cforest Univariate* and *cforest Teststatistic* are second fastest; finally *GLM forest* algorithms are the slowest. Let us denote that the **randomForest** package is well optimized with some routines in C



(a) Gamma $m = 20$

(b) Inverse Gaussian $m = 20$

(c) Gamma $m = 20$

(d) Inverse Gaussian $m = 20$

Figure 4: Error metrics as a function of `maxdepth`

and Fortran unlike **partykit** and that our implementation of *GLM forests* is developed in R. Therefore, the superiority of **randomForest** was expected since the different algorithms are not compared on the same playing field in terms of language. Yet, it should be noted that the greater complexity of **randomForest** leads to a larger number of operations, which means that an implementation of *GLM forests* in an equivalent language could lead to similar runtime performance.

4.4 Performance on `BostonHousing` and `Hitters` datasets

Finally, we assess the performance of *GLM forests*, *cforests* and *RandomForests* with `ntree=500`, `mtry=5`, `maxdepth=8` on `BostonHousing` and `Hitters`, see Section 3.5. As in Section 4.1, we consider three versions of *GLM forests* by choosing Gaussian, gamma and inverse Gaussian distributions (with canonical link) as for *GLM trees*. Regarding *cforests*, we restrict ourselves to only one version based on *Teststatistic* which is the most effective.

Figure 5 displays the predictive RMSE, whereas Table 11 displays the complexity and the runtime over 100 bootstrap replications. For `BostonHousing`, most complex forests produced by a *randomForest*

			Complexity		Computation time	
data	method	family	mean	median	mean	median
ContG2	cforest_Bonferroni	Gaussian	728.00	640.5	2.03	1.72
	cforest_Teststatistic	Gaussian	13244.31	13226.0	34.90	34.86
	cforest_Univariate	Gaussian	1775.54	1732.5	5.61	5.59
	glmforest	Gamma	1263.94	1224.5	219.71	219.28
	glmforest	Gaussian	1432.09	1388.0	177.48	181.01
	glmforest	Inverse Gaussian	1516.63	1504.0	176.86	178.98
	randomForest	gaussian	127905.45	127912.0	3.55	3.55
ContIG2	cforest_Bonferroni	Gaussian	845.10	794.0	5.39	4.20
	cforest_Teststatistic	Gaussian	15303.13	15202.0	82.62	61.72
	cforest_Univariate	Gaussian	2670.41	2567.0	18.35	14.38
	glmforest	Gamma	1451.66	1387.5	424.80	406.55
	glmforest	Gaussian	1587.87	1530.0	253.51	247.73
	glmforest	Inverse Gaussian	1480.79	1381.5	224.90	213.51
	randomForest	gaussian	127306.26	127334.0	7.18	5.75

Table 10: Complexity and runtime mean and median for `ContG2` and `ContIG2` over 100 runs, `maxdepth`=8

are the best followed by a *cforest*. Again there is no advantage of using non-Gaussian distributions for this dataset and the predictive performance for *GLM forests* is slightly worse than that of *cforests* and more unstable. For `Hitters`, RMSE boxplots of all algorithms are more close, yet *randomForests* perform slightly better at the price of complexity. It is thus interesting to note the parsimony of *cforests* and *GLM forests*.

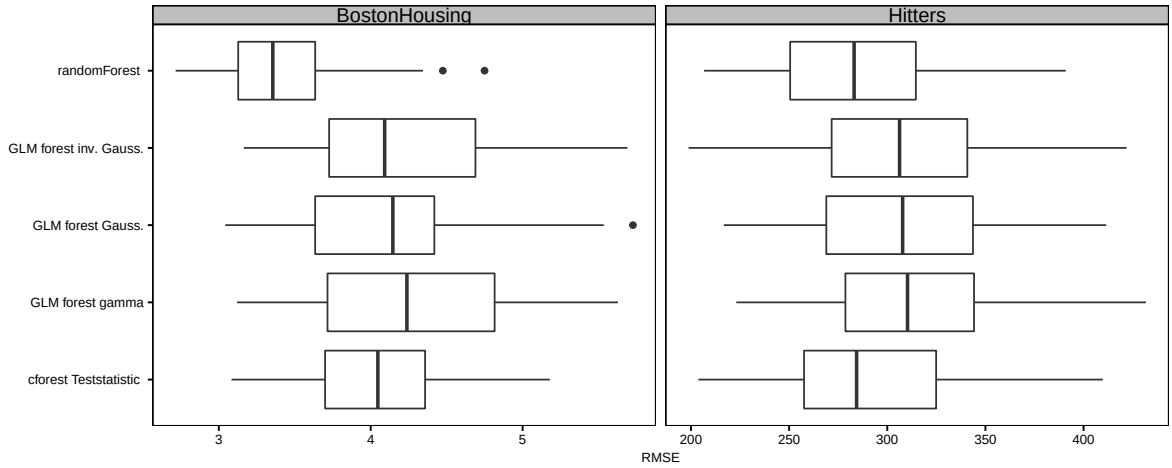


Figure 5: Predictive RMSE with 100 bootstrap replications for `BostonHousing` and `Hitters`

5 Conclusion

This paper focuses on GLM-based trees which are a particular case of MOB employed for partitioning GLM. This algorithm is of particular interest for identifying relevant subgroups in data where a GLM is locally estimated. We propose a new fast algorithm for growing GLM trees based on the use of explicit MLE solutions for GLM with categorical explanatory variables. The proposed algorithm in Section 2 is flexible and can be applied for any probability distribution in the one-para-meter exponential family and any link function. The main method presented in the paper relies on binary trees with intercept-only nodes, but we also show that it can be combined with multiway splits and the use of several explanatory

			Complexity		Computation time	
data	method	family	mean	median	mean	median
BostonHousing	cforest_Teststatistic	Gaussian	11569	11555	27.13	27.16
	glmforest	Gamma	6193	6184	258.50	259.00
	glmforest	Gaussian	6386	6396	231.98	231.95
	glmforest	Inverse Gaussian	6198	6194	228.38	227.93
	randomForest	Gaussian	62808	62818	0.88	0.88
Hitters	cforest_Teststatistic	Gaussian	6614	6606	16.48	16.48
	glmforest	Gamma	3134	3076	105.36	104.56
	glmforest	Gaussian	3291	3286	96.92	96.87
	glmforest	Inverse Gaussian	3161	3074	95.82	94.64
	randomForest	Gaussian	33153	33146	0.41	0.42

Table 11: Complexity and runtime mean and median for `BostonHousing` and `Hitters` over 100 runs

categorical variables in the GLM. Our approach also has explicit objective functions for a weighted MLE and transformed response variables.

We demonstrate on simulated and empirical datasets that this approach greatly increases the computation speed of the GLM-based tree model compared to the features originally offered by the R package **partykit**. Particularly, we show in Section 3 the proposed algorithm is three times faster for a Bernoulli response, five times faster for continuous non-Gaussian responses than the original. Furthermore, our numerical applications on continuous simulated datasets confirm the effective out-of-sample performance compared to other tree-based approaches such as *RPART* or *ctree*, both with categorical and continuous partitioning variables. Hence, this increases the interest of these models for applications where they are used intensively, as for ensemble decision trees. Despite of the use of closed-form estimators, our approach developed in R and based on the **partykit** package remains however much slower than `rpart` and `ctree` functions, indicating that additional efforts would be needed to speed up this method in C or Fortran code, both available in base R API.

Furthermore, this new algorithm makes it possible to derive a GLM forest algorithm in Section 4. Although this is also proposed by Garge et al. (2013), the use of closed-form estimators reduces the gaps in computation times to the other classical random forest algorithms such as *randomForests* and *cforests*. Similarly to *cforests*, *GLM forests* produce less complex trees than *randomForests* which could be interesting for the purpose of interpretability. For numerical applications based on simulated datasets, our GLM Forest approach performs better than its competitors in terms of MAE and is similar in terms of RMSE, depending on the chosen maximum depth of the tree. For the two chosen empirical datasets, GLM Forest does not perform better than classical random forest, but offers a good compromise in terms of performance and complexity.

This approach opens up some pathways for future research. Other types of distributions could be studied in the framework model-based trees, for instance inflated distributions such as zero-inflated Poisson, see, e.g., CORE models by Liu et al. (2019), two-parameter exponential families such as beta, negative binomial distributions or heavy-tailed distributions as in Farkas et al. (2021). In addition, we believe this method can be applied to other ensemble decision tree algorithm, such as boosted trees, or for prediction rule ensembles (Fokkema, 2020) where the features of MOBs if of interest for interpretable rule generation.

Acknowledgments

The authors are also very grateful for the useful suggestions of the two anonymous referees, which led to significant improvements of this article. The remaining errors, of course, should be attributed to the authors alone. This paper also benefits from fruitful discussions with members of the French chair

DIALog – Digital Insurance And Long-term risks – under the aegis of the Fondation du Risque, a joint initiative by UCBL and CNP; and with members of the French laboratory SAF (UCBL and ISFA).

Conflict of interest

The authors declare that they have no conflict of interest.

References

Box, G. and Cox, D. (1964). An analysis of transformations revisited. In: *Journal of American Statistician* 77, pp. 209–210 (cit. on p. 9).

Breiman, L. (Aug. 1996). Bagging Predictors. In: *Machine Learning* 24.2, pp. 123–140. URL: https://doi.org/10.1023/A:1018054314350 (cit. on p. 2).

Breiman, L. (Oct. 2001). Random Forests. In: *Machine Learning* 45.1, pp. 5–32. URL: https://doi.org/10.1023/A:1010933404324 (cit. on p. 2).

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (Jan. 1984). Classification and Regression Trees. New Ed. Boca Raton: Chapman and Hall/CRC (cit. on p. 1).

Brouste, A., Dutang, C., and Rohmer, T. (2020). Closed form Maximum Likelihood Estimator for Generalized Linear Models in the case of categorical explanatory variables: Application to insurance loss modelling. In: *Computational Statistics* 35, pp. 689–724 (cit. on pp. 2, 4, 6–11, 25).

Chambers, J. and Hastie, T. (1993). Statistical Models in S. Chapman and Hall (cit. on p. 8).

Ciampi, A. (Aug. 1991). Generalized regression trees. In: *Computational Statistics & Data Analysis* 12.1, pp. 57–78. URL: http://www.sciencedirect.com/science/article/pii/0167947391901039 (cit. on p. 2).

Cortes, C. and Vapnik, V. (1995). Support-vector networks. In: *Machine Learning* 20.3, pp. 273–297 (cit. on p. 2).

Denuit, M., Hainaut, D., and Trufin, J. (2019). Effective Statistical Learning Methods for Actuaries I: GLMs and extensions. Springer Actuarial Lecture Notes. Springer (cit. on pp. 8, 12).

Fahrmeir, L. and Kaufmann, H. (1985). Consistency and asymptotic normality of the maximum likelihood estimator in generalized linear models. In: *The Annals of Statistics*, pp. 342–368 (cit. on p. 3).

Farkas, S., Lopez, O., and Thomas, M. (May 2021). Cyber claim analysis using Generalized Pareto regression trees with applications to insurance. In: *Insurance: Mathematics and Economics* 98, pp. 92–105. URL: https://www.sciencedirect.com/science/article/pii/S0167668721000330 (cit. on p. 22).

Fokkema, M. (Mar. 2020). Fitting Prediction Rule Ensembles with R Package pre. In: *Journal of Statistical Software* 92.1, pp. 1–30. URL: https://www.jstatsoft.org/index.php/jss/article/view/v092i12 (cit. on pp. 2, 6, 22).

Friedman, J. H. (Feb. 2002). Stochastic gradient boosting. In: *Computational Statistics & Data Analysis*. Nonlinear Methods and Data Mining 38.4, pp. 367–378. URL: https://www.sciencedirect.com/science/article/pii/S0167947301000652 (cit. on p. 2).

Gama, J. (June 2004). Functional Trees. In: *Machine Learning* 55.3, pp. 219–250. URL: https://doi.org/10.1023/B:MACH.0000027782.67192.13 (cit. on pp. 2, 11).

Garge, N. R., Bobashev, G., and Eggleston, B. (Apr. 2013). Random forest methodology for model-based recursive partitioning: the mobForest package for R. In: *BMC Bioinformatics* 14, p. 125. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3626834/ (cit. on pp. 2, 22).

Hothorn, T., Hornik, K., and Zeileis, A. (Sept. 2006). Unbiased Recursive Partitioning: A Conditional Inference Framework. In: *Journal of Computational and Graphical Statistics* 15.3, pp. 651–674. URL: https://doi.org/10.1198/106186006X133933 (cit. on pp. 1, 2).

Hothorn, T. and Zeileis, A. (2015). partykit: A Modular Toolkit for Recursive Partytioning in R. In: *Journal of Machine Learning Research* 16, pp. 3905–3909. URL: https://jmlr.org/papers/v16/hothorn15a.html (cit. on pp. 2, 13).

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2017). ISLR: Data for an Introduction to Statistical Learning with Applications in R. URL: https://CRAN.R-project.org/package=ISLR (cit. on p. 16).

Kass, G. (1980). An exploratory technique for investigating large quantities of categorical data. In: *Annals of Applied Statistics* 29, pp. 119–127 (cit. on p. 10).

Kim, H. and Loh, W.-Y. (June 2001). Classification Trees With Unbiased Multiway Splits. In: *Journal of the American Statistical Association* 96.454, pp. 589–604. URL: https://doi.org/10.1198/016214501753168271 (cit. on pp. 1, 10).

Landwehr, N., Hall, M., and Eibe, F. (2005). Logistic Model Trees. In: *Machine Learning* 59, pp. 161–205 (cit. on p. 11).

Lawrence, J. (1994). Introduction To Neural Networks: Design, Theory and Applications. 6th. California Scientific Software (cit. on p. 2).

Leisch, F. and Dimitriadou, E. (2021). mlbench: Machine Learning Benchmark Problems. URL: https://CRAN.R-project.org/package=mlbench (cit. on p. 16).

Liaw, A. and Wiener, M. (2002). Classification and Regression by randomForest. In: *R News* 2.3, pp. 18–22. URL: https://CRAN.R-project.org/doc/Rnews/ (cit. on p. 18).

Liu, N.-T., Lin, F.-C., and Shih, Y.-S. (May 2019). Count regression trees. In: *Advances in Data Analysis and Classification*. URL: https://doi.org/10.1007/s11634-019-00358-7 (cit. on p. 22).

Loh, W.-Y. (2002). Regression trees with unbiased variable selection and interaction detection. In: *Statistica Sinica* 12.2, pp. 361–386. URL: https://www.jstor.org/stable/24306967 (cit. on p. 2).

Loh, W.-Y. (2014). Fifty Years of Classification and Regression Trees. In: *International Statistical Review* 82.3, pp. 329–348. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/insr.12016 (cit. on p. 1).

Loh, W.-Y. and Shih, Y.-S. (1997). Split Selection Methods for Classification Trees. In: *Statistica Sinica* 7.4, pp. 815–840. URL: https://www.jstor.org/stable/24306157 (cit. on p. 1).

Loh, W.-Y. and Vanichsetakul, N. (Sept. 1988). Tree-Structured Classification via Generalized Discriminant Analysis. In: *Journal of the American Statistical Association* 83.403, pp. 715–725. URL: https://amstat.tandfonline.com/doi/abs/10.1080/01621459.1988.10478652 (cit. on pp. 1, 11).

McCullagh, P. and Nelder, J. (Aug. 1989). Generalized Linear Models. Second Edition. Statistics and Applied Probability 37. Boca Raton: Chapman and Hall/CRC (cit. on pp. 3, 9).

Philipp, M., Rusch, T., Hornik, K., and Strobl, C. (Oct. 2018). Measuring the Stability of Results From Supervised Statistical Learning. In: *Journal of Computational and Graphical Statistics* 27.4, pp. 685–700. URL: https://doi.org/10.1080/10618600.2018.1473779 (cit. on p. 2).

R Core Team (2021). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. URL: http://www.R-project.org (cit. on pp. 2, 8).

Rusch, T. and Zeileis, A. (July 2013). Gaining insight with recursive partitioning of generalized linear models. In: *Journal of Statistical Computation and Simulation* 83.7, pp. 1301–1315. URL: https://doi.org/10.1080/00949655.2012.658804 (cit. on pp. 1–4).

Seber, G. A. and Lee, A. J. (2003). Linear regression analysis. John Wiley & Sons (cit. on p. 3).

Seibold, H., Hothorn, T., and Zeileis, A. (Oct. 2018). Generalised linear model trees with global additive effects. In: *Advances in Data Analysis and Classification*. URL: https://doi.org/10.1007/s11634-018-0342-1 (cit. on p. 1).

Su, X., Wang, M., and Fan, J. (Sept. 2004). Maximum Likelihood Regression Trees. In: *Journal of Computational and Graphical Statistics* 13.3, pp. 586–598. URL: https://amstat.tandfonline.com/doi/abs/10.1198/106186004X2165 (cit. on pp. 2, 7).

Szöcs, E. and Schäfer, R. B. (2015). Ecotoxicology is not normal: A comparison of statistical approaches for analysis of count and proportion data in ecotoxicology. In: *Environmental Science and Pollution Research* 22.18, pp. 13990–13999 (cit. on pp. 8, 12).

Therneau, T. and Atkinson, B. (2019). rpart: Recursive Partitioning and Regression Trees. URL: https://CRAN.R-project.org/package=rpart (cit. on p. 13).

Venables, W. and Ripley, B. (2002). Modern Applied Statistics with S. Springer (cit. on p. 7).

Weisberg, S. (2005). Applied linear regression. John Wiley & Sons (cit. on p. 3).

Wilson, K. and Grenfell, B. T. (1997). Generalized linear modelling for parasitologists. In: *Parasitology today* 13.1, pp. 33–38 (cit. on p. 8).

Wood, S. N. (2011). Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.1, pp. 3–36. URL: https://doi.org/10.1111/j.1467-9868.2010.00749.x (cit. on pp. 12, 26).

Zeileis, A. and Hornik, K. (2007). Generalized M-fluctuation tests for parameter instability. In: *Statistica Neerlandica* 61.4, pp. 488–508. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9574.2007.00371.x (cit. on pp. 2, 4, 5, 9).

Zeileis, A., Hothorn, T., and Hornik, K. (June 2008). Model-Based Recursive Partitioning. In: *Journal of Computational and Graphical Statistics* 17.2, pp. 492–514. URL: https://amstat.tandfonline.com/doi/abs/10.1198/106186008X319331 (cit. on pp. 1, 4, 6, 7, 12).

A Computation details on likelihood and deviance

A.1 Proof of the fitted log-likelihood

Using Corollary 3.1 of Brouste et al. (2020) for the j-th partitioning variable and using notations from Table 2, we have

$$\begin{aligned}
&\log L(\widehat{\theta}_L(s), \widehat{\theta}_R(s), \boldsymbol{y}, s) \\
= \; & \frac{1}{a(\phi)} \sum_{i \in L(j,s)} \left(y_i \tilde{b}\left(\overline{y}_j^L(s)\right) - b\left(\tilde{b}\left(\overline{y}_j^L(s)\right)\right)\right) + \frac{1}{a(\phi)} \sum_{i \in R(j,s)} \left(y_i \tilde{b}\left(\overline{y}_j^R(s)\right) - b\left(\tilde{b}\left(\overline{y}_j^R(s)\right)\right)\right) + \sum_{i=1}^{n} c(y_i, \phi) \\
= \; & \frac{1}{a(\phi)} \sum_{l \in \{L,R\}} \left(\tilde{b}\left(\overline{y}_j^l(s)\right) m_j^l(s) \overline{y}_j^l(s) - b\left(\tilde{b}\left(\overline{y}_j^l(s)\right)\right) m_j^l(s)\right) + \sum_{i=1}^{n} c(y_i, \phi),
\end{aligned}$$

where $\tilde{b} = (b')^{-1}$ is the inverse of b'.

A.2 Proof of the fitted deviance

Let us note the fitted mean $\hat{\mu}_i = g^{-1}(< z_i, \widehat{\boldsymbol{\theta}(s)} >) = \overline{y}_n^{(j)}$. The deviance is given by

$$\begin{aligned}
D(\widehat{\boldsymbol{\mu}}, \boldsymbol{y}) &= -2 \log L(\widehat{\boldsymbol{\mu}}, \phi, \boldsymbol{y}) + 2 \log L(\boldsymbol{y}, \phi, \boldsymbol{y}) \\
&= \frac{-2}{a(\phi)} \sum_{j=1}^{2} \tilde{b}\left(\overline{y}_n^{(j)}\right) \overline{y}_n^{(j)} m_j - \frac{-2}{a(\phi)} \sum_{j=1}^{2} b\left(\tilde{b}\left(\overline{y}_n^{(j)}\right)\right) m_j + \frac{2}{a(\phi)} \sum_{i=1}^{n} \tilde{b}\left(y_i\right) y_i - \frac{2}{a(\phi)} \sum_{i=1}^{n} b\left(\tilde{b}\left(y_i\right)\right).
\end{aligned}$$

A.3 Proof of the fitted log-likelihood

We adapt the proof of Brouste et al. (2020) in order to take into account weights when maximizing the log-likelihood. For a known weight w_i, their Equations (4) and (5) become

$$\log L(\boldsymbol{\theta} \,|\, \boldsymbol{y}) = \sum_{i=1}^{n} w_i \frac{y_i \ell(\eta_i) - b\left(\ell(\eta_i)\right)}{a(\phi)} + \sum_{i=1}^{n} w_i c(y_i, \phi),$$

$$S_j(\boldsymbol{\theta}) = \frac{1}{a(\phi)} \sum_{i=1}^{n} w_i x_i^{(j)} \ell'(\eta_i) \left(y_i - b'\left(\ell(\eta_i)\right)\right).$$

We adapt their proof of Theorem 3.1. The system $S(\boldsymbol{\theta}) = 0$ is

$$\begin{cases}
\displaystyle\sum_{i=1}^{n} \ell'(\eta_i) w_i \left(y_i - b' \circ \ell(\eta_i)\right) = 0 \\
\displaystyle\sum_{i=1}^{n} x_i^{(2),j} \ell'(\eta_i) w_i \left(y_i - b' \circ \ell(\eta_i)\right) = 0, \quad \forall j \in J.
\end{cases}$$

The first equation being redundant, the second equation simplifies to $\forall j \in J$

$$\ell'(\theta_{(1)} + \theta_{(2),j}) \sum_{i=1}^n x_i^{(2),j} w_i y_i$$
$$-\ell'(\theta_{(1)} + \theta_{(2),j}) b' \circ \ell(\theta_{(1)} + \theta_{(2),j}) \sum_{i=1}^n x_i^{(2),j} w_i = 0.$$

Using weighted frequencies $m_w^{(j)} = \sum_{i=1}^n x_i^{(2),j} w_i$, and averages $\overline{y}_w^{(j)} = \frac{1}{m_w^{(j)}} \sum_{i=1}^n x_i^{(2),j} w_i y_i$. Hence if y_i takes values in $\mathbb{Y} \subset b'(\Lambda)$, and ℓ injective, we have for all $j \in J$

$$\overline{y}_w^{(j)} = b' \circ \ell(\theta_{(1)} + \theta_{(2),j}) \Leftrightarrow \theta_{(1)} + \theta_{(j)} = g(\overline{y}_w^{(j)}).$$

The rest of the proof is identical except to replace $\overline{y}_n^{(j)}$ by $\overline{y}_w^{(j)}$. Hence, a slight modification of their Theorem 3.1 is for $j \in J$

$$\widehat{\boldsymbol{\theta}}_{n,(1)} = \frac{\sum_{k=1}^d r_k g(\overline{y}_w^{(k)})}{\sum_{k=1}^d r_k - r_0}, \widehat{\boldsymbol{\theta}}_{n,(2),j} = g(\overline{y}_w^{(j)}) - \frac{\sum_{k=1}^d r_k g(\overline{y}_w^{(k)})}{\sum_{k=1}^d r_k - r_0}.$$

For the two-variable case, we can also adapt their Theorem 3.2 by changing absolute frequencies and averages to weighted frequencies and averages as below.

Frequency	Mean	Index
$m_k^{(2)} = \sum_{i=1}^n x_i^{(2),k} w_i$	$\overline{y}_w^{(2),k} = \frac{1}{m_k^{(2)}} \sum_{i=1}^n y_i x_i^{(2),k} w_i$	$k \in K$
$m_l^{(3)} = \sum_{i=1}^n x_i^{(3),l} w_i$	$\overline{y}_w^{(3),l} = \frac{1}{m_l^{(3)}} \sum_{i=1}^n y_i x_i^{(3),l} w_i$	$l \in L$
$m_{k,l} = \sum_{i=1}^n x_i^{(k,l)} w_i$	$\overline{y}_w^{(k,l)} = \frac{1}{m_{k,l}} \sum_{i=1}^n y_i x_i^{(k,l)} w_i$	$(k,l) \in K \times L$

A.4 Proof of the log-likelihood of transformed variable

For $t(x) = \log(d_1 x + d_2)$, $t'(x) = \frac{d_1}{d_1 x + d_2}$, $t^{-1}(x) = (e^x - d_2)/d_1$ the log-likelihood of y such that $t(y)$ follows (1) is given by

$$f_Y(y) = f_Y(t(y)) t'(y) = \frac{d_1}{d_1 y + d_2} e^{\frac{\lambda t(y) - b(\lambda)}{a(\phi)} + c(y,\phi)} \Rightarrow \log f_Y(y) = \frac{\lambda t(y) - b(\lambda)}{a(\phi)} + c(y,\phi) + \log(\frac{d_1}{d_1 y + d_2}).$$

Therefore, the fitted log-likelihood based on (15) is

$$\log L_j(\hat{\theta}_L(s), \hat{\theta}_R(s), \boldsymbol{y}, s) = \frac{1}{a(\phi)} \sum_{l \in \{L,R\}} \tilde{b}\left(\overline{t}_{j,w}^l(s)\right) m_{j,w}^l(s) \overline{t}_{j,w}^l(s)$$
$$-\frac{1}{a(\phi)} \sum_{l \in \{L,R\}} b\left(\tilde{b}\left(\overline{t}_{j,w}^l(s)\right)\right) m_{j,w}^l(s) + \sum_{i=1}^n \tilde{c}(y_i, \phi), \tag{15}$$

where $t_{j,w}^l(s)$ are the corresponding observation of random variable $T_{j,w}^l(s)$ defined in Table 4.

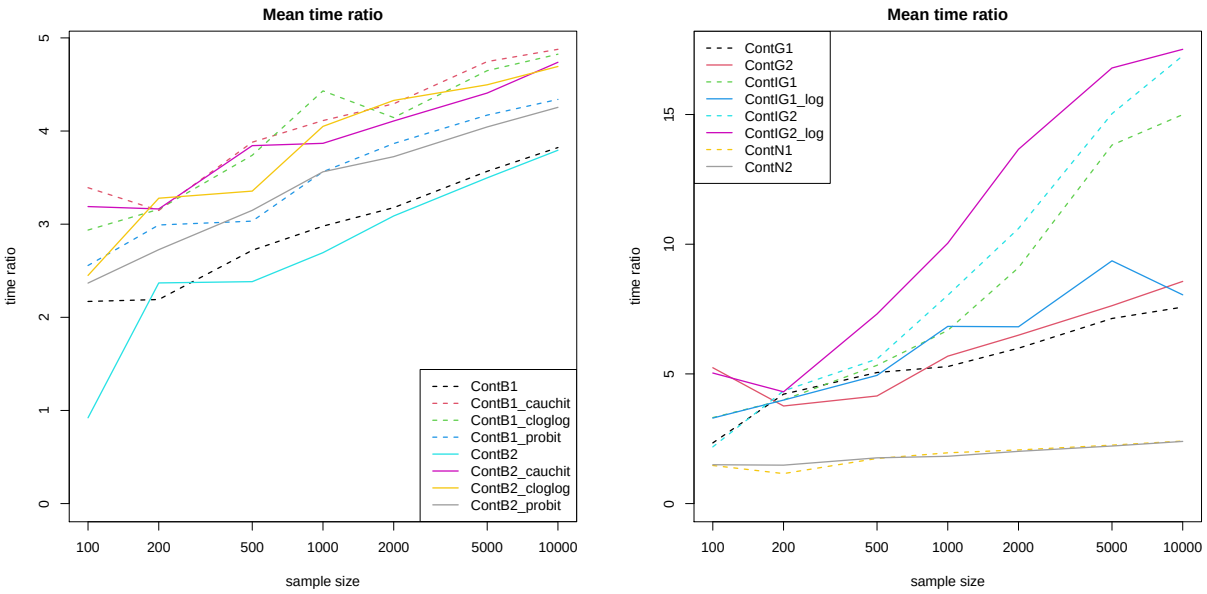
B Additional materials for numerical illustrations

B.1 Simon Wood's test functions

We define smooth functions to Simon Wood's test datasets (Wood, 2011) as

$f_0 = 5\sin(2\pi x)$, $f_1 = \exp(3x) - 7$, $f_2 = 0.5 \times x^{11}(10(1-x))^6 - 10(10x)^3(1-x)^{10}$,
$f_3 = 15\exp(-5|x - 1/2|) - 6$, $f_4 = 2 - 1_{(x<=1/3)}(6x)^3 - 1_{(x>=2/3)}(6-6x)^3 - 1_{(2/3>x>1/3)}(8 + 2\sin(9(x-1/3)\pi))$,
$f_5(x) = \lfloor 20x \rfloor - 10$, $f_6(x) = 10 - \lceil 20x \rceil$, $f_7(x) = \sin(50x) + 10x - 10$, $f_8(x) = 8 + 2\cos(50x) - 50x(1-x)$,
$f_9(x) = \lceil 50x(1-x) \rceil - 5$, $f_{10}(x) = 5\log(x + 10^{-6}) + 5$, $f_{11}(x) = -10 - 5\log(x + 10^{-6}) + \sin(50x)$,
$f_{12}(x) = 2\log(x + 10^{-6}) - 2\log(1 - x + 10^{-6})$, $f_{13}(x) = 10|\sin(20x)|$,
$f_{14}(x) = 1_{(x<=1/2)} \times 5\sin(20x) + 1_{(x>1/2)} \times (5\sin(10) + (\exp(5(x-0.5)) - 1))$.

B.2 Runtime comparison between *GLM tree* and *explicit GLM tree*



(a) Bernoulli distribution

(b) Continuous distributions

Figure 6: Mean runtime ratio of *GLM tree* over *explicit GLM tree*