BoXHED: Boosted eXact Hazard Estimator with Dynamic covariates

Xiaochen Wang [1] **Arash Pakbin** [2] **Bobak J. Mortazavi** [2] **Hongyu Zhao** [1] **Donald K.K. Lee** [3]

Abstract

The proliferation of medical monitoring devices makes it possible to track health vitals at high frequency, enabling the development of dynamic health risk scores that change with the underlying readings. Survival analysis, in particular hazard estimation, is well-suited to analyzing this stream of data to predict disease onset as a function of the time-varying vitals. This paper introduces the software package BoXHED (pronounced 'box-head') for nonparametrically estimating hazard functions via gradient boosting. BoXHED 1.0 is a novel tree-based implementation of the generic estimator proposed in Lee et al. (2017), which was designed for handling time-dependent covariates in a fully nonparametric manner. BoXHED is also the first publicly available software implementation for Lee et al. (2017). Applying it to a cardiovascular disease dataset from the Framingham Heart Study reveals novel interaction effects among known risk factors, potentially resolving an open question in clinical literature. BoXHED is available from GitHub: www.github.com/BoXHED.

1. Introduction

Driven by numerous applications in healthcare data analytics, interest in machine learning methods for survival data is rising steadily. These machine learning techniques often focus on classification problems (e.g. binary prediction of mortality in the next 72 hours), but we may be interested in not only the probability of an event occurring, but also in when that event might occur. For patients admitted to an Intensive Care Unit (ICU), for example, the high frequency nature of data collected by electronic health record systems allows us to prognosticate not only the chance of a patient dying, but also when that might happen, in order to provide timely critical care. Another example is the Framingham Heart Study, where longitudinal measurements allow us to ask not only if a patient will develop cardiovascular diseases (CVDs), but also when they are likely to develop them. In these settings, uncovering complex interactions of the dynamics of these risk factors could help quantify real-time risk and enable life-saving care. We explore an approach to this time-to-event analysis, called survival analysis in the statistics literature, to improve upon existing techniques by accounting for the time-varying nature of patient risk factors and potentially addressing new clinical findings.

Survival data describe the time to an event of interest (e.g. patient mortality), and a characteristic feature is the presence of censoring and/or truncation of the actual time-to-event T. This loss of information makes survival analysis fundamentally different from learning from continuous-valued data on the real line. For example, in lieu of the density $\mathbb{P}(T \in dt)$ and cumulative distribution $\mathbb{P}(T \leq t)$, the hazard $\lambda(t) = \mathbb{P}(T \in dt | T > t)$ and survivor function $S(t) = \mathbb{P}(T > t)$ are more natural for the survival setting. Recent works for survival analysis centre on methods for estimating the conditional survivor function

$$S(t|X) = \mathbb{P}(T > t|X) = \exp\left(-\int_0^t \lambda(u, X)du\right) \quad (1)$$

given time-static covariates $X \in \mathbb{R}^p$ that are fixed at $t = 0$.

In many real world settings, the covariates $X(t) \in \mathbb{R}^p$ can in fact change over time (e.g. patient temperature), and thanks to advances in technology, their readings can now be captured in real time. When $X(t)$ is time-varying there is no meaningful analogue to the survivor function (1), since it involves integrating the hazard $\lambda(u, X(u))$ along the unknown future trajectory of the covariates $\{X(u)\}_{u\in(0,t]}$. Thus in the time-dependent covariate setting, we are instead interested in estimating the hazard $\lambda(t, x)$ given $X(t) = x$, which is informally the probability of the event happening in the near future given it has not happened yet. Note that in the time-static covariate setting, we can recover $S(t|x)$ from $\lambda(t, x)$ via (1). Therefore, by focusing on hazard estimation we can unify the analyses of both settings.

However, nonparametric hazard estimation in the time-

[1]Biostatistics Department, Yale University, New Haven, Connecticut, USA [2]Computer Science & Engineering, Texas A&M University, College Station, Texas, USA [3]Goizueta Business School and Department of Biostatistics & Bioinformatics, Emory University, Atlanta, Georgia, USA. Correspondence to: Donald K.K. Lee <donald.lee@emory.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

dependent covariate setting is challenging because the covariate trajectories of the observations are functional data points. This may explain the sparsity of the literature on this topic. A recent work by Lee et al. (2017) examined this problem. Specifically, they proposed a functional gradient boosting algorithm for estimating the hazard nonparametrically using arbitrary weak learners. Theoretical guarantees are provided for the estimator, and they also outlined a prototype tree-based implementation as proof-of-concept. The authors deferred the development of a more refined implementation for future research, and hence did not provide software.

In order to design a scalable implementation that is clinically and software deployable, and to extend the machine learning toolbox for survival analysis to the time-dependent covariate setting, we introduce the algorithm and software package BoXHED (pronounced 'box-head'). This is the first publicly available implementation of the generic hazard estimator proposed in Lee et al. (2017). The current version (1.0) is written in Python and uses regression trees as learners. We first describe the novel algorithmic aspects of our implementation vis-à-vis the tree-based prototype in Lee et al. (2017). The key innovation of BoXHED is in developing a novel approach for growing trees that is more targeted at likelihood risk reduction, and as a result gives rise to a more direct variable importance measure. We evaluate the performance of BoXHED on simulation experiments, and also use it to analyze a cardiovascular disease dataset from the Framingham Heart Study. Of note, BoXHED discovered novel interaction effects among known risk factors in the Framingham data, potentially resolving an open question in the clinical literature. This further illustrates the utility of our tool for performing healthcare data analytics.

2. Related Work

The machine learning literature on survival analysis to date focus mainly on the time-static covariate setting (Ishwaran et al., 2008; Ranganath et al., 2016; Bellot & van der Schaar, 2018; Bellot & Schaar, 2019; Lee et al., 2019). There is, however, growing awareness of the importance of time-dependent covariates. For example, neural networks have been developed for hazard modeling in a discrete-time setting (Jarrett et al., 2018; Ren et al., 2019). This is equivalent to solving a series of binary classification problems, one at each point in an equally spaced time-grid. We note that there is also literature on neural network survival models that are not hazard-based, also for the discrete-time setting.

The extension to hazard modeling in a continuous-time setting has a longer history in the statistics literature. For time-static covariates, boosting has been applied to both parametric hazard models (Bühlmann & Hothorn, 2007) as well as the semiparametric Cox proportional hazards model (Ridgeway, 1999; Li & Luan, 2005; Binder & Schumacher, 2008). Where time-dependent covariates are concerned, kernel smoothing estimators (Nielsen & Linton, 1995; Pérez et al., 2013) have been proposed for low-dimensional covariate settings (along with theoretical guarantees), but otherwise the literature is rather sparse.

3. Problem Description

Consider functional datapoints collected from n units, with the i-th one represented by

$$(X_i(t)_{t\leq \tilde{T}_i}, \tilde{T}_i, \Delta_i),$$

where $\tilde{T}_i$ is the minimum of the event time T_i and censoring time C_i, $\Delta_i = 1$ if T_i is not censored ($T_i \leq C_i$) and $\Delta_i = 0$ otherwise, and $X_i(t) = \left(X_i^{(1)}(t), \cdots, X_i^{(p)}(t)\right)$ are covariates observed[1] from enrollment time ($t = 0$) up to $\tilde{T}_i$.

As a concrete example, suppose that each unit is a patient monitored in an ICU, and the event of interest is mortality in the ICU. A patient is censored if he/she is discharged alive, and $\tilde{T}$ is the minimum of the time-to-mortality and the time-to-discharge, as measured from the time of admission. The components of $X_i(t)$ could be blood pressure, heart rate, temperature, laboratory test results, and nursing assessments (Ma et al., 2019). The hazard $\lambda(t, X_i(t))$ in this case is informally the probability of dying in the ICU in the near future based on patient i's status $X_i(t)$ at time t.[2]

Writing $F(t, x) = \log \lambda(t, x)$ as the log-hazard function, BoXHED estimates $F(t, x)$ by using functional gradient boosting to minimize the negative log-likelihood functional (the likelihood risk)

$$R_n(F) = \frac{1}{n}\sum_{i=1}^{n}\left\{\int_0^{\tilde{T}_i} e^{F(t,X_i(t))}dt - \Delta_i F(\tilde{T}_i, X_i(\tilde{T}_i))\right\} \quad (2)$$

for F in the span of regression trees. Interestingly the functional gradient of $R_n(F)$ cannot be directly computed from (2) because of the way it depends on the functional datapoints $\{X_i(t)_{t\leq\tilde{T}_i}\}_{i=1}^n$. Lee et al. (2017) resolves this issue by proving a smooth convex representation for $R_n(F)$, and deriving its functional gradient for arbitrary learner classes. The computational algorithm for BoXHED described in the next section specializes these results to regression tree learn-

[1] If $X_i(t)$ is sampled at discrete (but possibly irregular) time points, BoXHED interpolates its trajectory with piecewise constant paths.

[2] It is also possible to use recent status history $\{X(s)\}_{s\in[t-\tau,t]}$ instead of just current status $X(t)$. See for example Adelson et al. (2017) and Ma et al. (2019) on how to transform status history into a time-dependent covariate.

ers in order to propose a novel approach for growing the trees.

4. The BoXHED Algorithm

Algorithm 1 describes the BoXHED algorithm for estimating $\lambda(t,x)$. As an overview, BoXHED creates a sequence $g_0(t,x), g_1(t,x), \cdots$ of regression tree functions of time t and covariates x in an iterative manner, and uses them to form an ensemble estimate for the log-hazard function

$$F_M(t,x) = F_0 - \nu \sum_{m=0}^{M-1} g_m(t,x),$$

from which the hazard estimator can be obtained as $\hat{\lambda}(t,x) = e^{F_M(t,x)}$. Here, M is the number of boosting iterations, and the default learning rate $\nu = 0.1$ is commonly used in boosting applications. The initial guess for the log-hazard, $F_0 = \log \frac{\sum_{i=1}^n \Delta_i}{\sum_{i=1}^n \tilde{T}_i}$, is the best constant that minimizes (2), where the ratio is the total number of observed events divided by the total amount of time at-risk. The number of boosting iterations M as well as the maximum number of splits L in each tree are hyperparameters that are chosen via K-fold cross-validation.

Given $F_m(t,x)$, the next iterate $F_{m+1}(t,x)$ is computed by seeking a regression tree $g_m(t,x)$ to add to $F_m(t,x)$, i.e. $F_{m+1} \leftarrow F_m - \nu g_m$. In gradient boosting, $g_m(t,x)$ needs to be aligned to the gradient function of $R_n(F)$ at F_m. This leads to a reduction in $R_n(F)$ from one iteration to the next. The following subsection describes how BoXHED constructs $g_m(t,x)$ to reduce $R_n(F)$ in a more direct manner than the prototype implementation in Lee et al. (2017).

4.1. Constructing the Tree $g_m(t,x)$

At the m-th iteration we seek a regression tree $g_m(t,x)$ that is aligned to the gradient. Since $R_n(F)$ is convex (Lee et al., 2017), the alignment property holds for any $g(t,x)$ that leads to a decrease in $R_n(F_m)$ when F_m is moved in the direction of $-g(t,x)$. Moreover, the larger the decrease is, the greater the alignment. This key insight behind BoXHED leads to a greedy approach for growing $g_m(t,x)$ that is similar to the spirit of XGBoost (Chen & Guestrin, 2016): Starting with a tree with a root node, we choose the split that maximally reduces the likelihood risk, and repeat the process iteratively on successive leaf nodes until the tree has been split (up to) L times. To explain further, let the tree with just a root node be the constant function $g_{m,0}(t,x) = 0$, and let

$$g_{m,l}(t,x) = \sum_{j=1}^{l+1} c_{m,j} I_{B_{m,j}}(t,x)$$

be the intermediate tree after l splits. Here, the splits can be on the covariates or on time. The indicator function $I_B(t,x)$

Algorithm 1 BoXHED

Input: n functional data samples $\{X_i(t)_{t\le\tilde{T}_i}, \tilde{T}_i, \Delta_i : i = 1,\dots,n\}$, #iterations M, maximum tree splits L, and default stepsize $\nu = 0.1$.
Initialize $F_0 = \log(\sum_{i=1}^n \Delta_i / \sum_{i=1}^n \tilde{T}_i)$.
for $m = 0$ **to** $M-1$ **do**
  Initialize $g_{m,0} = 0$.
  for $l = 1$ **to** L **do**
    Identify the tree split $(B_{m,j}, A_1, A_2)$ that minimizes the score d in (8).
    if $d < 0$ **then**
      Calculate (γ_1, γ_2) from (6).
      Update tree: $g_{m,l}(t,x) = g_{m,l-1}(t,x) - c_{m,j} I_{B_{m,j}}(t,x) + \gamma_1 I_{A_1}(t,x) + \gamma_2 I_{A_2}(t,x)$.
    else
      break
    end if
  end for
  Compute $F_{m+1} \leftarrow F_m - \nu g_m$.
end for
Output: $\hat{\lambda}(t,x) = e^{F_M(t,x)}$.

represents whether $(t,x) = (t, x^{(1)}, \cdots, x^{(p)})$ belongs in the time-covariate hypercube of the form

$$B = \left\{ (t,x) \quad : \quad \begin{array}{c} \underline{t}^B < t \le \overline{t}^B \\ \underline{x}^{(1,B)} < x^{(1)} \le \overline{x}^{(1,B)} \\ \vdots \\ \underline{x}^{(p,B)} < x^{(p)} \le \overline{x}^{(p,B)} \end{array} \right\}. \quad (3)$$

$B_{m,1}, \cdots, B_{m,l+1}$ are disjoint regions from the first l splits, each representing one of the leaf nodes in the intermediate tree. $c_{m,1}, \cdots, c_{m,l+1}$ are the values of the tree function in those leaf nodes. To obtain $g_{m,l+1}(t,x)$, we split one of the $B_{m,j}$ regions into two subregions A_1 and A_2 of the same form as (3) to get

$$g_{m,l+1}(t,x) = g_{m,l}(t,x) - c_{m,j} I_{B_{m,j}}(t,x) + \gamma_1 I_{A_1}(t,x) + \gamma_2 I_{A_2}(t,x).$$

The region $B_{m,j}$ to split, the variable or time axis to split on and the location of the split, and also the values of (γ_1, γ_2) are all chosen to minimize $R_n(F_m - g_{m,l+1})$.

Departure from the prototype implementation in Lee et al. (2017). The splits in Section 4 of Lee et al. (2017) are chosen to maximize the alignment between the tree and the gradient function of $R_n(F)$. As we know from traditional gradient boosting, subtracting a tree that is closely aligned to the gradient should reduce $R_n(F)$. By contrast, BoXHED splits are chosen to directly minimize $R_n(F)$, something that our novel algorithm makes possible, and this leads to more targeted risk reduction. Thus the difference between

the two implementations is analogous to the difference between XGBoost and the traditional boosting approach as they pertain to regression/classification problems. Given the performance gain XGBoost enjoys over the traditional approach, we expect BoXHED to also perform better than the prototype in Lee et al. (2017). We empirically validate this in Section 5.

Whereas XGBoost minimizes a second order Taylor approximation to a risk function in order to speed up computations, an innovation of BoXHED is in discovering how to directly minimize the exact form of $R_n(F)$ in an efficient way: Since the tree values γ_1, γ_2 only apply to the subregions A_1, A_2, we can write $R(F_m - g_{m,l+1})$ as

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{2}\Bigg\{\int_0^{\tilde{T}_i} e^{F_m(t,X_i(t))-\gamma_k} I_{A_k}(t,X_i(t))dt \\
&\quad - \Delta_i\left[F_m(\tilde{T}_i,X_i(\tilde{T}_i))-\gamma_k\right] I_{A_k}(\tilde{T}_i,X_i(\tilde{T}_i))\Bigg\} + C \\
=&\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{2}\Bigg\{e^{-\gamma_k}\cdot\int_0^{\tilde{T}_i} e^{F_m(t,X_i(t))} I_{A_k}(t,X_i(t))dt \\
&\quad + \gamma_k\cdot\Delta_i I_{A_k}(\tilde{T}_i,X_i(\tilde{T}_i))\Bigg\} + C'
\end{aligned}$$

where C, C' are quantities that do not depend on γ_1 or γ_2. Rewriting the above yields

$$R_n(F_m - g_{m,l+1}) = \sum_{k=1}^{2}[e^{-\gamma_k}U_k + \gamma_k V_k] + C', \quad (4)$$

$$\begin{aligned}
U_k &= \frac{1}{n}\sum_{i=1}^{n}\int_0^{\tilde{T}_i} e^{F_m(t,X_i(t))} I_{A_k}(t,X_i(t))dt, \\
V_k &= \frac{1}{n}\sum_{i=1}^{n}\Delta_i I[\{\tilde{T}_i, X_i(\tilde{T}_i)\}\in A_k].
\end{aligned} \quad (5)$$

Note that V_k is the (scaled) number of events observed in A_k. If $U_k, V_k > 0$ then the minimizing value in A_k is

$$\gamma_k = \log(U_k/V_k). \quad (6)$$

Putting this into the expression for $R(F_m - g_{m,l+1})$ above yields the optimized value

$$R(F_m - g_{m,l+1}) = \sum_{k=1}^{2} V_k\left(1+\log\frac{U_k}{V_k}\right) + C'. \quad (7)$$

By reasoning inductively, the decrease in the likelihood risk due to the new split is

$$\begin{aligned}
d &= R(F_m - g_{m,l+1}) - R(F_m - g_{m,l}) \\
&= V_1\left(1+\log\frac{U_1}{V_1}\right) + V_2\left(1+\log\frac{U_2}{V_2}\right) \\
&\quad - (V_1+V_2)\left(1+\log\frac{U_1+U_2}{V_1+V_2}\right),
\end{aligned} \quad (8)$$

which can be viewed as a score for determining the best split: Of all possible splits (defined by $B_{m,j}$, A_1, and A_2) where $U_k, V_k > 0$ in both subregions, the best one is that which minimizes (8). Since at each iteration only two new leaf nodes are created, it is only necessary to determine the best split for each of the two new regions at the next iteration: d remains unchanged for the other leaf nodes from previous iterations. If $d \geq 0$ for all leaf nodes then we stop splitting because doing so will not reduce the likelihood risk further.

One difference between growing a tree for the traditional non-functional data setting versus the functional data setting (our setting) is that for the latter, when choosing a variable to split on, time t is also treated as a candidate variable just like $x^{(1)}, \cdots, x^{(p)}$.

4.1.1. Candidate Split Points for Variables

Given a set of candidate split points for a particular variable, the best split point is that which minimizes (8). If $x^{(j)}$ is continuous, BoXHED proposes split candidates based on the percentiles of the observed data for $x^{(j)}$. The default setting places a candidate split at every decile.

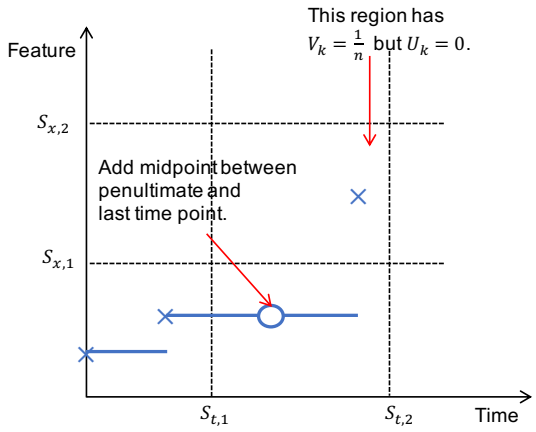
If $x^{(j)}$ is categorical, BoXHED employs a one-hot encoding heuristic: Set A_1 equal to the intersection of $B_{m,j}$ with the region where $x^{(j)}$ equals a particular categorical label. Hence A_2 is the intersection of $B_{m,j}$ with the region where $x^{(j)}$ is any other label. The algorithm would then choose the category label for A_1 that minimizes the score (8). The rationale for this is that if U_k and V_k can be varied continuously, then (8) tends to $-\infty$ as the ratio U_1/V_1 tends to ∞. Hence a heuristic is to find a subset of categorical labels to intersect with $B_{m,j}$ so that U_1/V_1 is maximized. This always has a solution in the form of a singleton set.

By contrast, the prototype implementation in Lee et al. (2017) chooses splits to minimize the mean squared error between the tree and gradient. As such, BoXHED's splitting rules for both continuous and categorical variables are more directly targeted at reducing $R_n(F)$.

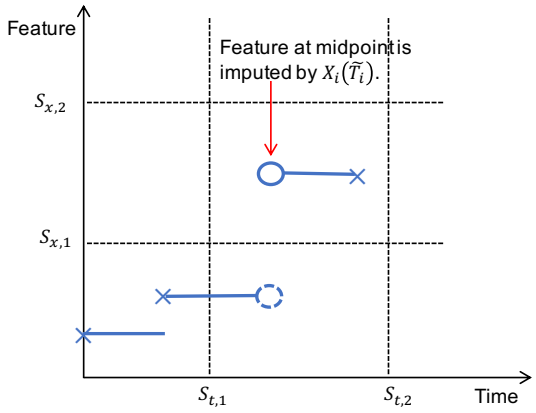
4.1.2. Imputing Covariate Trajectories

When the paths of $X(t)$ are continuous, no subregion A_k can contain an observed event unless it is traversed by at least one covariate trajectory, i.e. $V_k > 0 \Rightarrow U_k > 0$. This ensures that, as long as we consider only candidate splits where the number of observed events is positive in both subregions, $\gamma_k = \log(U_k/V_k)$ will be well defined.

However, when the paths of $X(t)$ are discontinuous, it is possible for $V_k > 0$ while $U_k = 0$. In Figure 1a, the trajectory jumps before time $S_{t,1}$ (the first split point on the time axis), and jumps again just before the second split $S_{t,2}$, whereupon the observation experiences the event at the rightmost $\times$. Here, the region bounded by $(S_{t,1}, S_{t,2})$



(a)



(b)

Figure 1. (a) Discontinuous covariate trajectory resulting in $V_k = 1/n$ and $U_k = 0$ for one of the time-covariate regions. (b) Imputed trajectory mitigates this problem.

on the time axis and by $(S_{x,1}, S_{x,2})$ on the covariate axis, has $V_k = 1/n$ but $U_k = 0$. To mitigate this, we impute a jump at the midpoint between the penultimate and the last time points (O in Figure 1b), and assign the values of the covariates from the last timepoint, $X_i(\tilde{T}_i)$, to the imputed one.

4.2. Defining Variable Importance

A variable importance measure can be constructed for the BoXHED estimator: Define the importance of the k-th variable (the zero-th one being time t) as

$$\mathcal{I}_k = \sum_{m=0}^{M-1} \mathcal{I}_k(g_m), \tag{9}$$

where for tree g_m with L internal nodes,

$$\mathcal{I}_k(g_m) = -\sum_{\ell=1}^{L} d_\ell I(v(\ell) = k) \geq 0.$$

Here, d_ℓ is the split score (8) at iteration ℓ and $v(\ell)$ is the variable used for the partition. Hence the second sum represents the total reduction in likelihood risk due to splits on the k-th variable in the m-th tree, and $\mathcal{I}_k$ is the total reduction across the M trees. To convert $\mathcal{I}_k$ into a measure of relative importance between 0 and 1, it is scaled by $\max_k \mathcal{I}_k$, where a larger value confers higher importance. Since the prototype implementation in Lee et al. (2017) fits trees to the gradients of $R_n(F)$ instead, it defines variable importance as the reduction in mean squared error between the trees and the gradients. Therefore, BoXHED's variable importance is a more direct measure of a variable's contribution to reducing the likelihood risk.

5. Numerical Study

We now compare the performance of BoXHED to those of several existing methods. For this, we use simulated datasets for which the true hazard function is known, thus allowing us to compute how well the methods do in recovering the truth. We then apply BoXHED to the Framingham Heart Study dataset to uncover a novel clinical finding concerning the relationship between blood pressure and CVD.

5.1. Performance Metrics

We evaluate the performance of the estimators on test data using two metrics: L^2-error and time-dependent AUC (AUC_t). Details on the training/test data splits can be found in Section 5.3.

The L^2-error is calculated on test datasets of N randomly sampled data points, and is defined as

$$\text{err}_{L^2} = \left\{ \frac{1}{N} \sum_{i=1}^{N} (\hat{\lambda}_i - \lambda_i)^2 \right\}^{1/2},$$

where $\hat{\lambda}_i$ and λ_i are the predicted and true hazard values for the i-th test data point. Note that err_{L^2} is always non-negative, and a smaller value indicates higher accuracy. In particular, $\text{err}_{L^2} = 0$ if and only if the predictions are perfect, i.e. all the predicted values are exactly the same as the true hazard values.

AUC_t is defined as follows (Blanche et al., 2019):

$$AUC_t = P\left(\hat{S}_i(t) < \hat{S}_j(t) | \Delta_i = 1, T_i < t < T_j\right),$$

where $\hat{S}_i(t)$ is the conditional survival probability given the covariate trajectory $\{X_i(s) : 0 \leq s \leq t\}$. Note that AUC_t lies between 0 and 1, and a value of 0.5 corresponds to a random guess. A larger value indicates better performance. However, AUC_t is not as sharp as err_{L^2} in detecting prediction mistakes. For example, if we overpredict by a factor of 2, i.e. $\hat{\lambda}_i = 2\lambda_i$, the corresponding AUC_t will still attain

Table 1. err_{L^2} with 95% confidence intervals for the simulated datasets (smaller values are better). Numbers rounded to two significant figures. BoXHED's hyper-parameters are tuned to the training data, whereas those for the other methods are tuned directly to the test data, Note that this puts BoXHED at an disadvantage. Furthermore, flexsurv includes the log-normal distribution as one of its parametric options, so it is correctly specified for λ_3.

Hazard	#Irrelevant covariates	Estimator			
		BoXHED	kernel	flexsurv	blackboost
λ_1	0	0.17 (0.17, 0.17)	**0.14 (0.14, 0.15)**	0.53 (0.52, 0.54)	0.58 (0.57, 0.59)
	20	**0.20 (0.20, 0.20)**	3.4 (3.0, 3.9)	0.54 (0.53, 0.54)	0.58 (0.57, 0.59)
	40	**0.21 (0.20, 0.21)**	43 (5.7, 80)	0.54 (0.54, 0.55)	0.58 (0.57, 0.59)
λ_2	0	0.23 (0.23, 0.24)	**0.11 (0.11, 0.12))**	1.1 (1.1, 1.1)	1.4 (1.4, 1.4)
	20	**0.25 (0.25, 0.26)**	4.5 (3.9, 5.2)	1.1 (1.1, 1.1)	1.4 (1.4, 1.4)
	40	**0.26 (0.26, 0.27)**	29 (11, 46)	1.1 (1.1, 1.1)	1.4 (1.4, 1.4)
λ_3	0	0.038 (0.037, 0.040)	0.046 (0.044, 0.049)	**0.0040 (0.0039, 0.0041)**	0.10 (0.10, 0.11)
	20	0.047 (0.046, 0.049)	1.8 (1.1, 2.5)	**0.020 (0.019, 0.020)**	0.10 (0.10, 0.11)
	40	0.050 (0.048, 0.051)	7.6 (5.3, 9.7)	**0.030 (0.029, 0.031)**	0.10 (0.10, 0.11)
λ_4	0	0.049 (0.048, 0.050)	**0.045 (0.044, 0.046)**	0.20 (0.19, 0.20)	0.20 (0.19, 0.20)
	20	**0.060 (0.059, 0.062)**	3.9 (0.66, 7.1)	0.20 (0.19, 0.20)	0.20 (0.19, 0.20)
	40	**0.069 (0.067, 0.070)**	5.5 (4.3, 6.7)	0.20 (0.20, 0.21)	0.20 (0.19, 0.20)

Table 2. BoXHED vs. the prototype implementation in Lee et al. (2017): Performance and speed comparisons. On average, BoXHED is 16% faster than the prototype and also achieves a 5.1% reduction in L^2-error. Numbers rounded to two significant figures.

Hazard	#Irrelevant covariates	Percentage reduction achieved by BoXHED	
		Error err_{L^2}	Computation time
λ_1	0	-0.35%	-0.52%
	20	3.7%	17%
	40	3.0%	0.72%
λ_2	0	1.9%	48%
	20	2.5%	-3.4%
	40	3.0%	16%
λ_3	0	5.0%	0.78%
	20	5.2%	1.1%
	40	5.0%	0.97%
λ_4	0	27%	57%
	20	1.8%	33%
	40	3.5%	15%
Average BoXHED improvement (95% C.I.)		**5.1%** (1.1%, 9.1%)	**16%** (4.0%, 27%)

the best possible value of 1, whereas err_{L^2} will strictly be larger than zero.

5.2. Baseline Comparisons

We compare BoXHED to several existing hazard estimation methods: Kernel smoothing (Nielsen & Linton, 1995), parametric hazard estimators for time-dependent covariates (flexsurv in R), and boosted parametric estimators for time-static covariates (blackboost in R). The Cox proportional hazards model is excluded because it is only able to estimate the cumulative hazard but not the hazard itself. The hyper-parameters[3] for BoXHED are tuned using five-fold cross-validation on the training data. For kernel smoothing we utilize the kernel function $K(u) = \frac{3003}{2048}(1-x^2)^6 I(-1 < x < 1)$ from Pérez et al. (2013). The hyper-parameters[4] for the baseline estimators are tuned directly to the test data (see Appendix for details). Note that this puts BoXHED at a significant disadvantage.

To assess the performance gain from using our novel tree splitting rule, we also compare BoXHED to the prototype implementation in Lee et al. (2017). Since no public implementation exists for the latter, we re-implemented a version of it to use for comparison.

[3] Candidates $L \in \{1, 2, 3, 4\}$ and $M \in \{100, 150, \cdots, 300\}$.

[4] Bandwidth for kernel, choice of parametric family for flexsurv and blackboost, and the number of trees in blackboost.

5.3. Data Description

Simulated datasets. We consider four datasets simulated from the following hazard functions used in Pérez et al. (2013), with x_t being a piecewise-constant function with values drawn from $U(0,1]$:

$$\lambda_1(t,x_t) = B(t,2,2) \times B(x_t,2,2), t \in (0,1],$$
$$\lambda_2(t,x_t) = B(t,4,4) \times B(x_t,4,4), t \in (0,1],$$
$$\lambda_3(t,x_t) = \frac{1}{t}\frac{\phi(\log t - x_t)}{\Phi(x_t - \log t)}, t \in (0,5],$$
$$\lambda_4(t,x_t) = \frac{3}{2}t^{\frac{1}{2}} \exp\left(-\frac{1}{2}\cos(2\pi x_t) - \frac{3}{2}\right), t \in (0,5],$$

where $B(\cdot,a,a)$ is the PDF of the Beta distribution with shape and scale parameters equal to a, and $\phi(\cdot)$ and $\Phi(\cdot)$ denote the PDF and CDF of $N(0,1)$. In other words, λ_1 and λ_2 take the form of Beta PDFs, and λ_3 is the hazard of the log-normal distribution. As will be explained in Section 5.4, the first two cases naturally favour `kernel` while the third one favours `flexsurv`.

To investigate the robustness of `BoXHED` to noise in a high dimensional setting, we also add up to 40 irrelevant covariates to each hazard function. The trajectories of the covariates are simulated as piecewise-constant paths with values drawn from $U(0,1]$.

For the training set we draw 5,000 sample trajectories, and we also draw 5,000 for the test set.

Framingham Heart Study dataset. We pool together longitudinal records from two prospective cohorts: The Framingham Heart Study original cohort (FHS) and the Framingham Heart Study Offspring Cohort (FHS-OS) (Dawber et al., 1951). The event of interest is the time to first onset of cardiovascular disease (CVD). We apply `BoXHED` to the data to identify important risk factors and to also uncover novel interaction effects. This might help us better understand the science behind CVD progression, and also to identify high-risk individuals for early intervention.

The pooled data consists of 9,697 participants and 73,340 physical exam records. Eight risk factors were consistently collected across all exams, and these are used in common medical models: Age, gender, systolic blood pressure (SBP), diastolic blood pressure (DBP), smoking status, diabetes, total cholesterol (TC), and body mass index (BMI). The 9,697 study participants are randomly split into 7,000/2,697 for training/testing. Additional details on cohort selection can be found in the Appendix.

5.4. Performance on Simulated Datasets

Table 1 presents the L^2-errors for the hazard estimators when applied to the simulated datasets. Several observations are in order:

- `BoXHED` always outperforms `kernel` when irrelevant covariates are present. The methods are comparable when no irrelevant covariates are present, with `kernel` having the edge in the first two cases λ_1 and λ_2. This is because the kernel function $K(u)$ is a location- and scale-transformed Beta PDF, which is also the functional form for λ_1 and λ_2. It is therefore unsurprising that `kernel` is able to approximate λ_1 and λ_2 better than regression trees. The minuscule edge that `kernel` enjoys for λ_4 is likely due to the fact that is was tuned directly to the test data.

- For λ_3, `flexsurv` performs the best, followed closely by `BoXHED`. The reason for `flexsurv`'s outperformance is due to the fact that it includes the log-normal distribution as one of its parametric options, so it is correctly specified for λ_3.

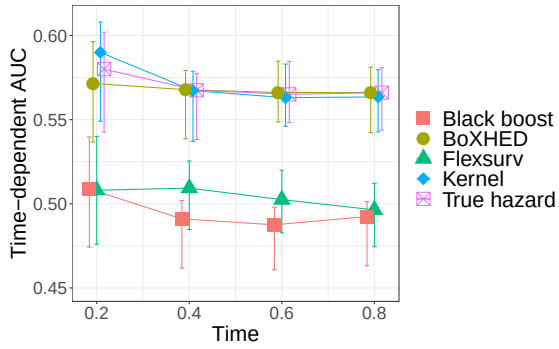
- Neither `BoXHED` nor `blackboost` are affected much by irrelevant covariates, while `kernel`'s performance drops dramatically when irrelevant covariates are added. These findings are in line with the fact that kernel smoothing suffers from the curse of dimensionality, while boosted trees automatically perform variable selection.

Figure 2a presents the AUC_t results for the estimators when applied to data simulated from λ_1 (no irelevant covariates). The performances of `BoXHED` and `kernel` are statistically indistinguishable from that of the true hazard function, while `blackboost` and `flexsurv` perform no better than random guessing. When we increase the number of irrelevant covariates to just 20 (Figure 2b), even `kernel` becomes statistically indistinguishable from random guessing. In terms of AUC_t, `BoXHED` remains as the only method that is able to perform as well as the true hazard. Similar results for λ_2, λ_3, and λ_4 are shown in the Appendix.

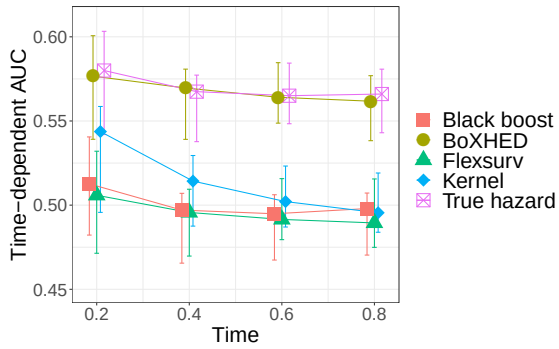
Table 2 compares the performance of `BoXHED` to that of the prototype implementation in Lee et al. (2017). We see that `BoXHED` outperforms the prototype in both speed and accuracy in 10 out of the 12 scenarios, and both approaches perform essentially the same on an eleventh one. On average, `BoXHED` achieved a 5.1% lower L^2-error (95% C.I. $1.1\% - 9.1\%$) and takes 16% less time to compute (95% C.I. $4.0\% - 27\%$), excluding the shared data pre-processing time.

5.5. Analyzing the Framingham Heart Study

We now use `BoXHED` to analyze the Framingham Heart Study dataset in order to identify important risk factors and interactions amongst them. Recall that the event of interest is the time to first onset of CVD.



(a)



(b)

Figure 2. AUC_t versus time t for the estimators when applied to data simulated from λ_1. Larger AUC_t values are better. (a) No irrelevant covariates; (b) 20 irrelevant covariates.

5.5.1. Variable Importances

Table 3 shows the relative variable importances (9) identified by BoXHED (scaled from 0 to 100). The top risk factors (age, SBP, smoking status, gender) match those identified by existing CVD prediction algorithms widely used by clinicians (e.g. ACA/AHA risk score (Arnett et al., 2019), Framingham risk score (Lloyd-Jones et al., 2004)).

5.5.2. Interactions Among Risk Factors

Since cross-validation selected trees with two splits, this suggests the presence of two-way interactions among the risk factors and time. We investigate the impact of blood pressure (BP) on CVD risk (defined as the hazard of CVD onset), and show that BoXHED is able to uncover clinically novel interaction effects.

To systematically examine the relationship between BP and CVD risk, we calculate the hazard estimated by BoXHED for 576 sub-cohorts. A sub-cohort is defined by combinations of 7 variables: Age (50, 60, and 70 years old), gender (women, men), TC (5, 5.7, 6.5 $mmol/L$), current smoker (yes, no), diabetes (yes, no), BMI (20, 25, 30, and 35 kg/m^2), and cohort indicator (FHS, FHS-OS). For each sub-cohort, the estimated hazard is computed for different values of BP:

Table 3. Relative importances (out of 100) and 95% confidence intervals (CI) for the risk factors in the Framingham Heart Study. Confidence intervals obtained from 1,000 bootstrapped datasets.

Age	100
SBP	15 (11,21)
Smoking	9.5 (6.3,12)
Gender	8.3 (5.8,11)
Diabetes	6.2 (4.1,8.8)
DBP	3.5 (2.5,8.4)
Total Cholesterol	2.8 (2.1,6.0)
BMI	1.4 (1.2,3.5)

Table 4. Distribution of risk factors by clusters.

Risk Factor	Clusters			
	#1	#2	#3	#4
Age	60	60	60	60
Female (%)	**60**	**52**	**59**	**0**
Current Smoker (%)	50	48	51	50
Diabetes (%)	100	16	46	50
TC ($mmol/L$)	5.73	5.72	5.74	5.73
BMI (kg/m^2)	**29.0**	**23.4**	**27.1**	**35.0**

SBP $\in [90, 170]$ mmHg and DBP $\in [60, 110]$ mmHg. The estimated hazard surfaces are scaled to $[0, 1]$ and are aggregated into four clusters using K-means clustering (see Appendix for details).

Figure 3 exhibits the relationship between SBP and the scaled hazard for the centroid of each cluster, while also conditioning on DBP at three different values (75, 90, and 100 mmHg). Interestingly, for Clusters 1 to 3, CVD risk is increasing in SBP for a given level of DBP, while for Cluster 4 the relationship is U-shaped. In order to better understand the underlying interaction effect, in Table 4 we summarize the characteristics of the four clusters. Age, %smokers, and TC levels are similar across all clusters, while %diabetes are similar in Clusters 3 and 4. Gender and BMI are the only covariates that differentiate Cluster 4 from the others, and they indicate an all-male cluster with high BMI values. Therefore, we hypothesize that SBP$\times$BMI and/or SBP$\times$Gender might be the interaction effects responsible for the observed relationship between SBP and CVD risk.

We use logistic regression to test these hypotheses using odds ratios (ORs). ORs quantify the likelihood of developing CVD relative to the baseline cohort (defined here as SBP < 115 mmHG and DBP < 70 mmHG). An OR greater than 1 indicates that CVD onset is relatively more likely for the cohort of interest, whereas an OR less than 1 indicates that the cohort is relatively less risky. Observations from all participants are pooled together to evaluate the eight risk factors, SBP $\times$ DBP, and the candidate interaction effects

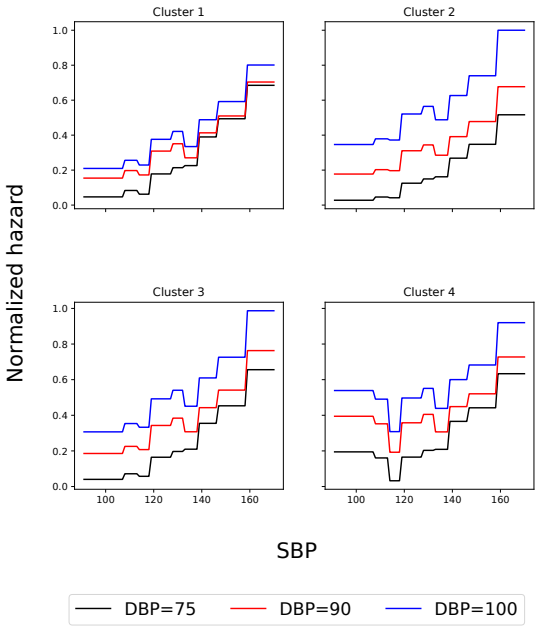


Figure 3. Plots of hazard against SBP for centroids of clusters.

SBP×BMI and SBP×Gender. SBP is bucketed into quintiles (<115, 115-124, 125-139, 140-149, and ≥150 mmHg), and DBP is bucketed in the same way (<70, 70-79, 80-84, 85-89, ≥90 mmHg).

Table 5 shows the ORs for the nine lower deciles of BMIs ($\leq 32kg/m^2$) as well as for the top decile ($> 32kg/m^2$). A similar stratification for SBP and Gender can be found in the Appendix. Cells containing fewer than 10 events are left blank since we do not have enough data to estimate the OR. Among the bottom nine deciles of BMIs (Table 5a), reading down a given column reveals an increasing relationship between OR and SBP for a given level of DBP. On the other hand, the top BMI decile (Table 5b) demonstrates a U-shaped relationship, with the minimum OR attained when the SBP is between 115 to 124 mmHg. This matches the trough seen in the plot for Cluster 4 in Figure 3. Repeating the analysis for SBP×Gender did not reveal a similar qualitative difference (see Appendix).

This line of inquiry, spurred by the findings of BoXHED, suggests that the SBP×BMI interaction effect might be responsible for the observed differences in the qualitative relationship between CVD risk and SBP. This potentially resolves an open question in the clinical literature regarding the differential impact of BP on CVD development among different patient cohorts (Herrington et al., 2017; Franklin & Wong, 2013). Indeed, the fact that prior work remove BMI as an explanatory variable (primary and interaction) may be precisely why the literature report seemingly contradictory findings on these relationships. While our analysis is not intended to be confirmatory, it provides clinical researchers with a concrete hypothesis on which to base further confirmatory analysis.

Table 5. ORs with 95% confidence intervals.

(a) Bottom nine deciles of BMIs ($\leq 32kg/m^2$).

SBP \ DBP	<70	70-79	80-84	85-89	>90
<115	1.0	0.9±0.2			
115-124	1.5±0.3	1.3±0.2	1.2 ±0.3		
125-139	1.8±0.3	1.5±0.3	1.3±0.2	1.9±0.3	1.8±0.4
140-149	2.0±0.5	2.0±0.4	1.7±0.4	2.0±0.4	2.3±0.6
>150	2.8±0.6	2.6±0.6	2.4±0.4	2.2±0.6	2.7±0.6

(b) Top decile of BMIs ($> 32kg/m^2$).

SBP \ DBP	<70	70-79	80-84	85-89	>90
<115	2.0±0.7	1.8±0.7			
115-124	1.4±0.5	1.2±0.4	1.1±0.5		
125-139	2±0.5	1.6±0.6	1.5±0.4	2.1±0.5	2±0.5
140-149	2±0.6	2±0.5	1.7±0.5	2.1±0.7	2.4±0.6
>150	3.3±1.0	3±0.7	2.8±0.7	2.6±0.6	3.1±0.6

6. Conclusion

Survival data with time-dependent covariates is becoming more prominent within the machine learning for healthcare community. As the first public implementation of a boosted nonparametric hazard estimator for time-dependent covariates, BoXHED's utility should increase as high frequency medical data becomes more prevalent. With an in-depth analysis of a heart study cohort, we demonstrate that i) BoXHED's built-in variable selection capabilities make it robust to high-dimensional data with many irrelevant covariates; ii) BoXHED is able to flexibly capture novel interaction effects among risk factors, potentially resolving an open clinical question; and iii) The information conveyed by the changes in risk factors over time may be valuable in prognosticating the onset of CVD. We have made BoXHED available to extend the machine learning toolbox for survival analysis.

Acknowledgements

BJM was supported in part by NIH grant 1R21EB028486-01. We are grateful to Huajie Qian for helpful discussions.

References

Adelson, K., Lee, D. K. K., Velji, S., Ma, J., Lipka, S. K., Rimar, J., Longley, P., Vega, T., Pérez-Irizarry, J., Pinker, E., and Lilenbaum, R. Development of imminent mortality predictor for advanced cancer (IMPAC), a tool to predict short-term mortality in hospitalized patients with

advanced cancer. *Journal of Oncology Practice*, 14(3): e168–e175, 2017.

Arnett, D. K., Blumenthal, R. S., Albert, M. A., Michos, E. D., Buroker, A. B., Miedema, M. D., Goldberger, Z. D., Muñoz, D., Hahn, E. J., Smith, S. C., et al. 2019 ACC/AHA guideline on the primary prevention of cardiovascular disease. *Journal of the American College of Cardiology*, pp. 26029, 2019.

Bellot, A. and Schaar, M. Boosting transfer learning with survival data from heterogeneous domains. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 57–65, 2019.

Bellot, A. and van der Schaar, M. Boosted trees for risk prognosis. In *Machine Learning for Healthcare Conference*, pp. 2–16, 2018.

Binder, H. and Schumacher, M. Allowing for mandatory covariates in boosting estimation of sparse high-dimensional survival models. *BMC Bioinformatics*, 9(1):14, 2008.

Blanche, P., Kattan, M. W., and Gerds, T. A. The c-index is not proper for the evaluation of-year predicted risks. *Biostatistics*, 20(2):347–357, 2019.

Bühlmann, P. and Hothorn, T. Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science*, pp. 477–505, 2007.

Chen, T. and Guestrin, C. XGBoost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. ACM, 2016.

Dawber, T. R., Meadors, G. F., and Moore Jr, F. E. Epidemiological approaches to heart disease: the framingham study. *American Journal of Public Health and the Nations Health*, 41(3):279–286, 1951.

Franklin, S. S. and Wong, N. D. Hypertension and cardiovascular disease: Contributions of the Framingham Heart Study. *Global heart*, 8(1):49–57, 2013.

Herrington, W., Staplin, N., Judge, P. K., Mafham, M., Emberson, J., Haynes, R., Wheeler, D. C., Walker, R., Tomson, C., Agodoa, L., et al. Evidence for reverse causality in the association between blood pressure and cardiovascular risk in patients with chronic kidney disease. *Hypertension*, 69(2):314–322, 2017.

Ishwaran, H., Kogalur, U. B., Blackstone, E. H., and Lauer, M. S. Random survival forests. *The Annals of Applied Statistics*, 2(3):841–860, 2008.

Jarrett, D., Yoon, J., and van der Schaar, M. MATCH-Net: Dynamic prediction in survival analysis using convolutional neural networks. *arXiv preprint arXiv:1811.10746*, 2018.

Lee, C., Zame, W., Alaa, A., and Schaar, M. Temporal quilting for survival analysis. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 596–605, 2019.

Lee, D. K. K., Chen, N., and Ishwaran, H. Boosted nonparametric hazards with time-dependent covariates. *arXiv preprint arXiv:1701.07926*, 2017.

Li, H. and Luan, Y. Boosting proportional hazards models using smoothing splines, with applications to high-dimensional microarray data. *Bioinformatics*, 21(10): 2403–2409, 2005.

Lloyd-Jones, D. M., Wilson, P. W., Larson, M. G., Beiser, A., Leip, E. P., D'Agostino, R. B., and Levy, D. Framingham risk score and prediction of lifetime risk for coronary heart disease. *The American Journal of Cardiology*, 94 (1):20–24, 2004.

Ma, J., Lee, D. K. K., Perkins, M. E., Pisani, M. A., and Pinker, E. Using the shapes of clinical data trajectories to predict mortality in ICUs. *Critical Care Explorations*, 1 (4):e0010, 2019.

Nielsen, J. P. and Linton, O. B. Kernel estimation in a nonparametric marker dependent hazard model. *Annals of Statistics*, 23(5):1735–1748, 1995.

Pérez, M. L. G., Janys, L., Martinez-Miranda, M. D., and Nielsen, J. P. Bandwidth selection in marker dependent kernel hazard estimation. *Computational Statistics and Data Analysis*, 68:155–169, 2013.

Ranganath, R., Perotte, A., Elhadad, N., and Blei, D. Deep survival analysis. In *Machine Learning for Healthcare Conference*, pp. 101–114, 2016.

Ren, K., Qin, J., Zheng, L., Yang, Z., Zhang, W., Qiu, L., and Yu, Y. Deep recurrent survival analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4798–4805, 2019.

Ridgeway, G. The state of boosting. *Computing Science and Statistics*, pp. 172–181, 1999.