



# A survival tree based on stabilized score tests for high-dimensional covariates

Takeshi Emura<sup>a</sup>, Wei-Chern Hsu<sup>b</sup> and Wen-Chi Chou<sup>c</sup>

<sup>a</sup>Biostatistics Center, Kurume University, Kurume, Japan; <sup>b</sup>Graduate Institute of Statistics, National Central University, Taoyuan, Taiwan; <sup>c</sup>Department of Hematology and Oncology, Chang Gung Memorial Hospital and College of Medicine, Chang Gung University, Taoyuan, Taiwan

## ABSTRACT

A survival tree can classify subjects into different survival prognostic groups. However, when data contains high-dimensional covariates, the two popular classification trees exhibit fatal drawbacks. The logrank tree is unstable and tends to have false nodes; the conditional inference tree is difficult to interpret the adjusted  $P$ -value for high-dimensional tests. Motivated by these problems, we propose a new survival tree based on the stabilized score tests. We propose a novel matrix-based algorithm in order to tests a number of nodes simultaneously via stabilized score tests. We propose a recursive partitioning algorithm to construct a survival tree and develop our original R package *uni.survival.tree* (<https://cran.r-project.org/package=uni.survival.tree>) for implementation. Simulations are performed to demonstrate the superiority of the proposed method over the existing methods. The lung cancer data analysis demonstrates the usefulness of the proposed method.

## ARTICLE HISTORY

Received 31 October 2020  
Accepted 3 October 2021

## KEYWORDS


CART; censoring;  
classification tree; gene  
expression; logrank test;  
prognostic prediction

## 1. Introduction

A classification tree is a tree-like model developed by a sequence of binary classifications (called splits) of the samples. A classification tree consists of internal nodes of the binary splits (branches) and terminal nodes of the classified groups (leaves). One of the major purposes of classification trees is to predict the outcome for a sample by identifying the most appropriate group. The tree provides a graphical tool for prediction, which is especially useful for clinicians/physicians to predict the disease outcomes for their patients.

Breiman *et al.* [5] described the classification tree as a non-linear regression model, which becomes a popular idea in statistics and machine learning community. In the statistical literature, the deviance and the two-sample test are the two most commonly used measures to find the optimal splitting rules. See Everitt and Howell [21] for the overview of the classification tree for continuous and discrete outcomes. For survival outcomes, which

**CONTACT** Takeshi Emura  [takeshiemura@gmail.com](mailto:takeshiemura@gmail.com)  Biostatistics Center, Kurume University, 67 Asahi-machi, Kurume, Japan

 Supplemental data for this article can be accessed here. <https://doi.org/10.1080/02664763.2021.1990224>

are typical outcomes for fatal diseases, the two most commonly used measures are the significance tests based on the logrank statistic and Cox regression analysis.

The earliest works of Ciampi *et al.* [9,10] used a logrank test for classifications, including the applications to non-Hodgkin's lymphoma patients and breast cancer patients. While the literature shows a large number of criteria to replace the logrank test [4,31,38], the classification tree based on the logrank test remains the simplest for computation and interpretation. However, the practical application of a tree is feasible only by the software packages.

So far, the R package '*rpart*' (Therneau and Atkinson [39]) is one of the most popular and well-developed tools to construct trees, along with the graphical add-on R package '*partykit*' (Hothorn *et al.* [27]). These powerful packages allow various response types, including continuous, binary, and survival responses. For survival responses, significance tests under the conditional Cox model are employed for selecting the optimal splits. The resultant tree is called 'conditional inference tree' (Hothorn *et al.* [25,26]), which is often abbreviated as '*ctree*'. Illustrations of the *ctree* are found in the well-known textbook of R (Hothorn and Everitt [24]).

High-dimensional covariates often arise while developing a survival prognostic prediction method with gene expressions [17,19,41,42]. Such data pose many challenges for users of a survival tree. When one applies the R package '*rpart*' to high-dimensional gene expression data, no covariate is deemed significant in the usual scaling of  $P$ -values (e.g. 0.01 and 0.05) due to the adjustment for multiple tests. It should be emphasized that the usual scaling of  $P$ -values (e.g. 0.01 and 0.05) for gene screening remains useful without adjustment (e.g. Beer *et al.* [3]; Chen *et al.* [7]; Zhang *et al.* [44]). It is also too time-consuming to find the optimal tree in the presence of a number of covariates. Therefore, the univariate score tests have been suggested to select significant covariates [19,34,35,42], which do not require model fitting and are workable for the commonly used  $P$ -value thresholds. A predictor constructed from the selected genes is useful for predicting survival in lung cancers [3,7], ovarian cancer [20], and other cancers [30,8] for the usual scaling of the  $P$ -value threshold, such as 0.05, 0.01, and 0.001.

Motivated by the computational efficiency and interpretability (of  $P$ -value) in the univariate score tests, this article introduces a new tree construction algorithm of classifying survival data with the high-dimensional covariates. This proposed method differs from the conditional tree based on the multivariate Cox-regression (Hothorn *et al.* [25]) that is implemented in the existing R packages. However, the methods are similar in its principle to use ' $P$ -value' as the threshold for determining the final tree. The proposed method also differs from the tree constructed by the logrank tests (called the *logrank tree*). We argue that the logrank tests are occasionally unstable and invalid for finding the best split in high-dimensional and small sample settings. The key novelty in the proposed method is the stabilization technique originally proposed by Witten and Tibshirani [42] for high-dimensional multiple tests. The stabilized score test is a minor correction to the logrank test, which however produces a large influence on a tree via multiple tests.

This article is constructed as follows. Section 2 reviews the background of the research. Section 3 proposes a new survival tree method. Section 4 includes the simulation studies. Section 5 contains a real data analysis. Section 6 concludes the article. Software implementation is given in Appendix.

## 2. Background

This section reviews the methods for classification trees for survival data. In order to motivate our newly proposed method, we point out some defects of the existing classification tree methods.

### 2.1. Notations

Let  $\{(t_i, \delta_i, \mathbf{x}_i); i = 1, 2, \dots, n\}$  be a survival dataset, where  $t_i = \min\{T_i, U_i\}$  is survival or censoring time such that survival time  $T_i$  and censoring time  $U_i$  are random variables,  $\delta_i$  is the censoring indicator ( $\delta_i = 1$  if  $t_i$  is survival time, or  $\delta_i = 0$  if  $t_i$  is censoring time),  $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})'$  is a  $p$ -dimensional vector of covariates, and  $n$  is the number of samples. Define a hazard function  $h(t|w_{ij}) = -d\log P(T_i > t|w_{ij})/dt$ , the instantaneous risk of death at  $t$  given the covariate information  $w_{ij}$ . The  $w_{ij}$  can be either  $\{x_{ij} > c\}$  or  $\{x_{ij} \leq c\}$  for a cut-off value  $c$ .

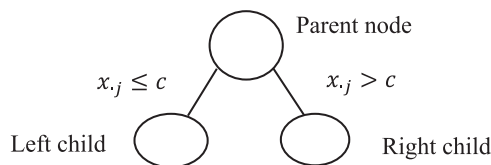
### 2.2. Survival trees

Survival trees are supervised classification algorithms to develop prognostic models based on observed survival outcomes. A tree constructed from survival data can classify patients into different prognostic groups according to their risk of death. With a survival tree, physicians and clinicians can intuitively classify their patients to identify their survival prognoses.

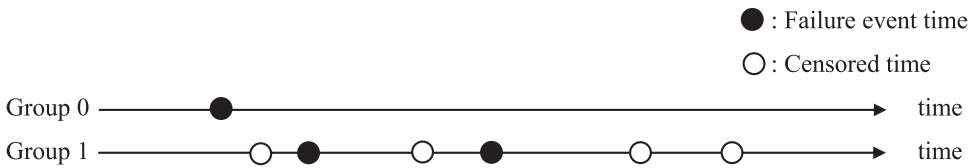
A survival tree is constructed by an algorithm called ‘recursive partitioning’. It is an algorithm to sequentially partition the samples into subgroups based on the data  $\{(t_i, \delta_i, \mathbf{x}_i); i = 1, 2, \dots, n\}$ . To construct a tree, we define a node by the split of the samples into two subgroups:  $\{i; x_{ij} \leq c\}$  and  $\{i; x_{ij} > c\}$ , where  $c$  is a cut-off-value. The choices for the covariate  $j$  and the value of  $c$  are determined by maximizing the prognostic difference between two subgroups. The set  $\{1, 2, \dots, n\}$  is called a parent node (or root node). Two sets of partitioned samples are called child nodes (Figure 1). A tree arises by recursively creating children from parents until some stopping criterion is met. Terminal nodes are the nodes without children. Obviously, different trees arise from different measures of the prognostic difference of the two child nodes.

### 2.3. Logrank tree

The logrank test has been the most commonly used way to measure the difference of the two groups in terms of survival prognosis. For constructing a survival tree, Ciampi



**Figure 1.** A split of a parent node by a covariate  $x_j$  and a cut-off value  $c$ .



**Figure 2.** Unbalanced sample allocation: one sample in Group 0 and other samples in Group 1.

*et al.* [10] proposed the logrank test as a criterion for splitting a parent node into two child nodes. Since then, it has been routinely utilized for constructing a survival tree [4,31,38]. The logrank tree minimizes the  $P$ -value for testing  $H_0 : h(t|x_j \leq c) = h(t|x_j > c)$  for each node in the tree. The recursive partitioning is terminated when the  $P$ -value gets larger than a pre-specified threshold.

However, the test results of the logrank test may lose the statistical meaning when the two child nodes have unusual sample allocations, the cases frequently occur in a tree construction algorithm for high-dimensional covariates. Here we give an example to describe the invalid results for the logrank test due to unbalanced sample sizes.

Suppose that one sample is allocated to Group 0 and other  $n - 1$  to Group 1 (Figure 2).

In this setting, the  $Z$ -value for testing the two groups' difference is

$$z = \frac{(\text{Observed} - \text{Expected})}{\sqrt{\text{Var}}} = \left(1 - \frac{1}{n}\right) / \sqrt{\frac{n-1}{n^2}} = \sqrt{n-1},$$

and the  $P$ -value =  $\Pr(|Z| > z)$  where  $Z \sim N(0, 1)$ . Thus, the  $P$ -value  $< 0.05$  if  $n \geq 5$ . However, such a small  $P$ -value may not indicate the true difference of the two groups. More generally, if one group contains the earliest  $k$  deaths, and the other group contains the remaining samples, the  $Z$ -value is

$$z = \frac{(\text{Observed} - \text{Expected})}{\sqrt{\text{Var}}} = \left\{k - \left(\frac{k}{n} + \dots + \frac{1}{n-k+1}\right)\right\} / \sqrt{\frac{k(n-k)}{n^2} + \dots + \frac{n-k}{(n-k+1)^2}} \approx \frac{k\sqrt{n}}{k+1}.$$

Again, it is likely to get the  $P$ -value  $< 0.05$  if  $n \geq 5$ . The false  $P$ -values generally arise in small samples, not restricted to the above cases.

In order to avoid the above mentioned problems in the logrank tree, Leblanc and Crowley [31] applied a permutation test. However, the permutation test is not a realistic approach for high-dimensional covariates due to its extremely high computational cost. A more common approach is to set the minimum number of samples in each terminal nodes in order to insure that the sample sizes in  $\{i; x_{ij} > c\}$  and  $\{i; x_{ij} \leq c\}$  are greater than a specified number (e.g. 10). However, the sample size itself does not have statistical meaning, difficult to interpret, and unclear by censoring. Thus, the choice of the sample size is often subjectively or arbitrary chosen by a user. Or, a user may try different choices before getting a tree that he/she wants. A bit more systematic way is to prune the tree after a large tree is constructed under a very small sample size threshold. This strategy typically needs to choose a

cost function and perform an additional cross-validation to optimize the prediction accuracy, which is extremely time-consuming for high-dimensional covariates. Our strategy to correct the logrank test is different from these complex or unrealistic approaches.

**2.4. Stabilized score test**

It is well-known that the logrank test is related to a score test under the Cox model  $h_j(t|w_{ij}) = h_{0j}(t)\exp(\beta_j w_{ij})$ , where  $w_{ij} = I(x_{ij} > c)$ , and  $h_{0j}(\cdot)$  is the baseline hazard function. Let  $R_i = \{\ell; t_\ell \geq t_i\}$  be the risk set at time  $t_i$ . For testing  $H_{0j} : \beta_j = 0$  vs.  $H_{1j} : \beta_j \neq 0$ , the score statistic and its variance are derived from the log-partial likelihood  $\ell_j(\beta_j)$  under  $\beta_j = 0$ ,

$$\begin{aligned} \ell_j(\beta_j) &= \sum_{i=1}^n \delta_i \left[ \beta_j w_{ij} - \log \left\{ \sum_{\ell \in R_i} \exp(\beta_j w_{\ell j}) \right\} \right] \\ S_j &= \left. \frac{\partial \ell_j(\beta_j)}{\partial \beta_j} \right|_{\beta_j=0} = \sum_{i=1}^n \delta_i \left[ w_{ij} - \sum_{\ell \in R_i} w_{\ell j} / \sum_{\ell \in R_i} 1 \right] \\ V_j &= \left. -\frac{\partial^2 \ell_j(\beta_j)}{\partial \beta_j^2} \right|_{\beta_j=0} = \sum_{i=1}^n \delta_i \left[ \sum_{\ell \in R_i} w_{\ell j}^2 / \sum_{\ell \in R_i} 1 - \left\{ \sum_{\ell \in R_i} w_{\ell j} / \sum_{\ell \in R_i} 1 \right\}^2 \right] \end{aligned}$$

for  $j = 1, \dots, p$ . The Z-value of the test is  $z_j = S_j/V_j^{1/2}$ , which is equivalent to the logrank test. Hence, the score and logrank tests share the same problem under unusual sample allocations.

Witten and Tibshirani [42] stabilized the Z-value by using a small constant  $d_0 \geq 0$  such that

$$z_j^{d_0} = S_j / (V_j^{1/2} + d_0).$$

The value  $d_0$  is regarded as a shrinkage parameter: a larger value  $d_0$  shrinks the Z-value more toward zero (Table 1). The value  $d_0 = 0$  corresponds to the original score statistic or the logrank statistic. Since  $V_j^{1/2} / (V_j^{1/2} + d_0)$  converges in probability to one, the asymptotic distribution of  $z_j^{d_0}$  is equivalent to that of the logrank test. This means that a P-value can be computed with the reference to the standard normal distribution as in the logrank test.

Table 1 shows that the stabilized Z-value mitigates the problem of the logrank test under the unbalanced sample allocation. Emura *et al.* [19] developed the *compound.Cox* R package that implements the computation of the stabilized Z-values in multiple tests. The function *uni.score(.)* in this package can efficiently compute many Z-values of the stabilized score test.

**2.5. Conditional inference tree (ctree)**

The *ctree* of Hothorn *et al.* [25] performs a recursive partition according to the tests for the hazard ratio via the conditional Cox model given the status of their ancestors. They select

**Table 1.** Z-values of the two-sample tests when the earliest  $k$  death occurs in one group.

	$k$	Z-value*	$n = 10$	$n = 50$
Logrank test (= score test)	1	2.000	3.000	7.000
	2	2.065	3.320	8.027
Stabilized score test with $d_0 = 0.01$	1	1.951	2.903	6.533
	2	2.034	3.256	7.708
Stabilized score test with $d_0 = 0.1$	1	1.600	2.250	4.083
	2	1.791	2.775	5.677

\* $z_j^{d_0} = S_j / (V_j^{1/2} + d_0)$ , where  $d_0 \geq 0$ . The value  $d_0 = 0$  corresponds to the logrank test.

a covariate  $x_j$  and its cut-off value  $c$  such that the conditional test for  $H_0 : h(t|x_j \leq c) = h(t|x_j > c)$  leads to the greatest significance. The most important advantage, as advocated in Hothorn *et al.* [25], is that the statistically meaningful measure – the  $P$ -value, can control the tree size without the sample size constraint and the pruning process. However, the interpretational difficulty arises when the dimension  $p$  is large (Section 5).

### 3. Proposed method

This section proposes a new method for constructing a survival tree based on the stabilized score test. Recall that the stabilized score test mitigates the problem of the logrank test under unbalanced samples between two splitting groups (Section 2.4). Hence, the new algorithm is intended to improve upon the logrank tree. Moreover, we explain how this algorithm is efficiently performed for the purpose of constructing a survival tree.

#### 3.1. Algorithm to find the optimal split

A tree is constructed by recursively partitioning a parent node into two children by finding the optimal split. Hence, it is essential to develop an efficient algorithm to test a number of hypotheses for all the possible splits. Below, we explain how efficiently the stabilized score tests are performed.

We wish to decide an optimal split for testing

$$H_{0j} : h(t|w_j = 0) = h(t|w_j = 1) \quad \text{vs.} \quad H_{1j} : h(t|w_j = 0) \neq h(t|w_j = 1),$$

where  $w_j = I(x_j > c)$ ,  $j = 1, 2, \dots, p$ . It is then necessary to search over all possible covariates  $x_j$ 's and their cut-offs by testing a number of hypotheses. Below, we propose an efficient computing method for performing multiple tests and finding the optimal split.

Let  $c_1 < c_2 < \dots < c_m$  be the  $m$  ordered values such that  $c_1$  is the smallest value and  $c_m$  is the largest value of  $x_{ij}$ 's. Thus, all  $x_{ij}$ 's are scaled into  $c_1 \leq x_{ij} \leq c_m$  so that  $c_1, c_2, \dots, c_{m-1}$  are the possible cut-off values for all the covariates, giving binary splits of  $\{i; x_{ij} \leq c_k\}$  vs.  $\{i; x_{ij} > c_k\}$  for  $k = 1, 2, \dots, m - 1$ . We then define a matrix of  $n \times p(m - 1)$  by

$$\begin{bmatrix} I(x_{11} > c_1) & I(x_{11} > c_2) \cdots I(x_{11} > c_{m-1}) \cdots I(x_{1p} > c_1) & I(x_{1p} > c_2) \cdots I(x_{1p} > c_{m-1}) \\ I(x_{21} > c_1) & I(x_{21} > c_2) \cdots I(x_{21} > c_{m-1}) \cdots I(x_{2p} > c_1) & I(x_{2p} > c_2) \cdots I(x_{2p} > c_{m-1}) \\ \vdots & \vdots & \vdots \\ I(x_{n1} > c_1) & I(x_{n1} > c_2) \cdots I(x_{n1} > c_{m-1}) \cdots I(x_{np} > c_1) & I(x_{np} > c_2) \cdots I(x_{np} > c_{m-1}) \end{bmatrix}$$

The  $(i, j)$  element of this matrix is denoted by  $w_{ij}$ ,  $i = 1, 2, \dots, n$ , and  $j = 1, 2, \dots, p(m - 1)$ . Then, for a model  $h_j(t|w_{ij}) = h_{0j}(t)\exp(\beta_j w_{ij})$ , we perform a test for  $H_{0j} : \beta_j = 0$ . The univariate score statistic and its variances are

$$S_j = \sum_{i=1}^n \delta_i(w_{ij} - S_{ij}^{(1)}/S_{ij}^{(0)}), \quad V_j = \sum_{i=1}^n \delta_i(S_{ij}^{(2)}/S_{ij}^{(0)} - (S_{ij}^{(1)}/S_{ij}^{(0)})^2),$$

where  $S_{ij}^{(r)} = \sum_{\ell \in R_i} w_{\ell j}^r$  for  $r \in \{0, 1, 2\}$ . For  $d_0 > 0$ , we obtain a  $Z$ -value  $z_j^{d_0} = S_j/(V_j^{1/2} + d_0)$  for  $j = 1, 2, \dots, p(m - 1)$ . Finally, the optimal split is determined by  $\operatorname{argmax}_j\{|z_j^{d_0}|\}$ .

To find the optimal split, it is necessary to perform high-dimensional tests on  $p(m - 1)$  covariates, which is computationally expensive. A naive method is to use the ‘for loop’ to compute them. However, consider a  $k$ -node tree having to compute over  $k \cdot p(m - 1)$  tests. Then, one requires a huge computational burden due to triple loops for  $k, p$ , and  $m$ . To overcome the problems of high-dimensional score tests, we extend the matrix-based computation technique of Emura *et al.* [19] to a tree algorithm. In order to apply their method, we first prepare the stacked matrices:

$$\begin{matrix} & \underbrace{\hspace{10em}}_{\times (m - 1)} & & \underbrace{\hspace{10em}}_{\times (m - 1)} \\ \mathbf{X} = & \begin{bmatrix} x_{11}x_{11} \cdots x_{11} & \vdots & \vdots & x_{1p}x_{1p} \cdots x_{1p} \\ \vdots & \vdots & \cdots & \vdots \\ x_{n1}x_{n1} \cdots x_{n1} & \vdots & \vdots & x_{np}x_{np} \cdots x_{np} \end{bmatrix} & \cdot \\ \mathbf{C} = & \begin{bmatrix} c_1c_2 \cdots c_{m-1} & \vdots & \vdots & c_1c_2 \cdots c_{m-1} \\ \vdots & \vdots & \cdots & \vdots \\ c_1c_2 \cdots c_{m-1} & \vdots & \vdots & c_1c_2 \cdots c_{m-1} \end{bmatrix} & \cdot \end{matrix}$$

Then, we define the binary split indicator matrix  $\mathbf{W} = I(\mathbf{X} > \mathbf{C})$ . We define a vector of score statistics and a vector of their variances as

$$\mathbf{S} = \boldsymbol{\delta}'(\mathbf{W} - \mathbf{S}^{(1)}/S^{(0)}), \quad \mathbf{V} = \boldsymbol{\delta}'(\mathbf{S}^{(2)}/S^{(0)} - (\mathbf{S}^{(1)}/S^{(0)}) \times (\mathbf{S}^{(1)}/S^{(0)})),$$

where  $\boldsymbol{\delta} = (\delta_1, \dots, \delta_n)'$  and

$$\mathbf{S}^{(r)} = \begin{bmatrix} S_{11}^{(r)}S_{12}^{(r)} \cdots S_{1(p(m-1))}^{(r)} \\ S_{21}^{(r)}S_{22}^{(r)} \cdots S_{2(p(m-1))}^{(r)} \\ \vdots \\ S_{n1}^{(r)}S_{n2}^{(r)} \cdots S_{n(p(m-1))}^{(r)} \end{bmatrix}, \quad r \in \{0, 1, 2\}.$$

Thus, the vector of  $Z$ -statistics is  $\mathbf{S}/(\mathbf{V}^{1/2} + d_0\mathbf{1})$ . Here, the operators  $\times$  and  $/$  are applied in the component-wise manner.

Above operations should be done on the vector **S** and **V** instead of computing  $S_j$  and  $V_j$  one-by-one through the loop for  $j = 1, 2, \dots, p(m - 1)$ . For R users, one first prepares **W**, and then applies it to the `uni.score(d0=)` function in the `compound.Cox` package. The output gives a vector of  $Z$ -values of dimension  $p(m - 1)$ . The following example illustrate how this idea works.

**Example 3.1:** (a toy dataset of  $n = 10$ )

We give an example of using the proposed computing method. Consider a dataset of size  $n = 10$ :

$t_j$	$\delta_j$	$X_{j1}$	$X_{j2}$	$X_{j3}$	$X_{j4}$	$X_{j5}$
1	0	2	1	2	4	4
2	0	1	3	2	3	1
3	1	2	3	3	1	1
4	0	3	2	3	2	4
5	1	4	2	3	3	4
6	0	3	1	2	4	3
7	1	4	1	4	1	3
8	1	4	2	2	2	2
9	0	1	4	1	3	2
10	1	1	4	1	2	4

A split indicator matrix defining all the possible splits is shown as follows:

```
>W
      x1>1 x1>2 x1>3 x2>1 x2>2 x2>3 x3>1 x3>2 x3>3 x4>1 x4>2 x4>3 x5>1 x5>2 x5>3
[1,] 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0
[2,] 0 0 0 1 1 0 1 0 0 1 1 0 0 0 0 0
[3,] 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0
[4,] 1 1 0 1 0 0 1 1 0 1 0 0 1 1 1 1
[5,] 1 1 1 1 0 0 1 1 0 1 1 0 1 1 1 1
[6,] 1 1 0 0 0 0 1 0 0 1 1 1 1 1 0 0
[7,] 1 1 1 0 0 0 1 1 1 0 0 0 1 1 0 0
[8,] 1 1 1 1 0 0 1 0 0 1 0 0 1 0 0 0
[9,] 0 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0
[10,] 0 0 0 1 1 1 0 0 0 1 0 0 1 1 1 1
```

We put this ‘**W**’ matrix into the function `uni.score(.)` in the `compound.Cox` package, which produces the vector of  $Z$ -values and  $P$ -values of the score tests:

```
> library(compound.Cox)
> uni.score(t.vec,d.vec,W)
.
.
$Z
      x1>1      x1>2      x1>3      x2>1      x2>2      x2>3      x3>1
1.8634487 0.9079092 1.3206444 -0.2156655 -0.9079092 -1.8634487 1.8634487
      x3>2      x3>3      x4>1      x4>2      x4>3      x5>1      x5>2
2.3597502 0.6943138 -1.8599622 -0.4847179 -0.5853694 -2.6457513 -0.1297013
      x5>3
-0.3133630

$P
      x1>1      x1>2      x1>3      x2>1      x2>2      x2>3
0.062399168 0.363926211 0.186619961 0.829248459 0.363926211 0.062399168
      x3>1      x3>2      x3>3      x4>1      x4>2      x4>3
0.062399168 0.018287244 0.487485411 0.062890871 0.627876457 0.558299355
      x5>1      x5>2      x5>3
0.008150972 0.896802746 0.754004863
```



We see that  $I(x_5 > 1)$  leads to the greatest significant since the  $P$ -value of the score test is the smallest, serving as the optimal split.

To see the effect of the stabilized score test, we set  $d_0 = 0.1$ . Then, the greatest significance is shown to be  $I(x_3 > 2)$ . Thus, the optimal split has been changed by stabilization.

```
> uni.score(t.vec,d.vec,W,d0=0.1)
.
$Z
      x1>1      x1>2      x1>3      x2>1      x2>2      x2>3      x3>1
1.6841190  0.8225595  1.1981422 -0.1909560 -0.8225595 -1.6841190  1.6841190
      x3>2      x3>3      x4>1      x4>2      x4>3      x5>1      x5>2
2.1010717  0.6029717 -1.6322658 -0.4383586 -0.4875242 -2.0314873 -0.1175085
      x5>3
-0.2829620

$P
      x1>1      x1>2      x1>3      x2>1      x2>2      x2>3      x3>1
0.09215867  0.41075852  0.23086165  0.84856009  0.41075852  0.09215867  0.09215867
      x3>2      x3>3      x4>1      x4>2      x4>3      x5>1      x5>2
0.03563467  0.54652753  0.10262351  0.66112636  0.62588688  0.04220559  0.90645710
      x5>3
0.77720600
```

**Remark 3.1:** The proposed algorithm is suitable to deal covariates having the same scale (the example above). However, clinical covariates, such as tumor size, gender, and cancer stages, may have different scales. The simplest way to unify the scales is to transform all covariates into quantile levels, e.g. quartile levels, 1, 2, 3, or 4 ( $\sim 25$ th, 25th  $\sim$  50th, 50th  $\sim$  75th, or 75th  $\sim$  percentile). A factorial covariate has to be transformed into either an ordinal covariate or dummy variables. A variable taking 0 and 1 has to be changed into 1 and 4 to conform this scale. In practice, the choice of  $c_1 < c_2 < \dots < c_m$  may not need a large  $m$ . Clinical interpretations of the cut-off values  $c_1, c_2, \dots, c_{m-1}$  may also be relevant.

### 3.2. Algorithm to construct a survival tree

We have suggested our optimal splitting strategy by minimizing the  $P$ -value for each split. After recursively splitting nodes, one needs to stop splitting when certain conditions are met. We suggest our stopping rule to be ‘ $P$ -value’, unlike a commonly used rule based on the number of samples in a node (or uncensored samples in a node). We believe that  $P$ -values are statistically more meaningful than sample sizes. Especially for censored data, the sample size carries little information. For instance, a node with 10 uncensored samples is more informative than a node with 10 censored samples. If a tree is determined by the sample size criterion, it has to perform an additional pruning process. Furthermore, users have little sense to determine how a sample size is large or small. On the other hand, they may agree that ‘ $P < 0.05$ ’, ‘ $P < 0.01$ ’, and ‘ $P < 0.001$ ’ are reasonable ‘evidence’ to have a split for a node. In summary, our survival tree adopts a single splitting/stopping criterion, the  $P$ -value, without restricting the node sizes or pruning a tree. This results in a very simple algorithm with an interpretable tree.

To develop our tree-construction algorithm, the initial step is to:

**Step 0:** Set a  $P$ -value threshold, denoted as  $0 < P < 1$  (e.g.  $P = 0.05$ ), and a shrinkage parameter  $d_0 > 0$ . Define a current (parent) node by all the samples in the dataset  $\{1, 2, \dots, n\}$ .

We propose the following recursive partitioning to construct a survival tree:

**Step 1:** If the current node has  $n \leq 2$ , set it as a terminal node. For  $n > 2$  in the current node, calculate the  $Z$ -value  $z_{jk}^{d_0}$  for testing  $\{i; x_{ij} \leq c_k\}$  vs.  $\{i; x_{ij} > c_k\}$  for all possible splits  $(j, k)$ 's. Find  $(j^*, k^*) = \operatorname{argmin}_{j,k} \{|P_{j,k}^{d_0}|\}$ , where  $P_{j,k}^{d_0} = \Pr(|Z| > |z_{j,k}^{d_0}|)$ . Set  $P_{j^*k^*} = \Pr(|Z| > |z_{j^*,k^*}^{d_0}|)$ .

**Step 2:** If  $P_{j^*k^*} < P$ , create two child nodes by splitting the current node into  $\{i; x_{ij^*} \leq c_{k^*}\}$  and  $\{i; x_{ij^*} > c_{k^*}\}$ : go back to **Step 1**, replacing the current node with the child nodes. If  $P_{j^*k^*} \geq P$ , stop the algorithm and define the current node as the terminal node.

**Remark 3.2:** If the samples in the current node are all censored, one has  $P_{j,k}^{d_0} = 1$  due to  $z_{jk}^{d_0} = 0$  for all possible splits  $(j, k)$ 's. Thus, this node becomes a terminal node no matter how the sample size is large. If the current node has a high proportion of censored samples, it is likely to be a terminal node. However, we stick to the  $P$ -value for the stopping rule without referring to the censored proportion. This is because the  $P$ -value contains every information necessary to judge the decision on the binary split.

### 3.3. Selection of $d_0$

We suggest minimizing the Akaike Information Criterion (AIC) to choose  $d_0$  for a given  $P$ -value threshold. The AIC of a tree can be computed from a Cox model having the terminal nodes as factorial covariates (computed by the  $AIC(\cdot)$  function in R). If the tree has unbalanced samples in the terminal nodes, the Cox model may not converge. Thus, the value of  $d_0$  is optimized among those models making the Cox model converges. As  $d_0$  increases, the likelihood tends to decrease while the number of terminal nodes decreases. The AIC achieves a good balance between the convergence, goodness-of-fit, and the model complexity. We do not recommend cross-validation that requires a high cost of computation.

Different values of  $d_0$  can produce the same AIC value because there are a limited number of trees coming from all the values of  $d_0$ . In our data example, the values  $d_0 = 0$ ,  $d_0 = 0.01$ ,  $d_0 = 0.1$ , and  $d_0 = 0.2$  were enough to produce all possible trees. Among them,  $d_0 = 0.01$  was chosen as the best choice while  $d_0 = 0$  and  $d_0 = 0.1$  did not make their Cox models converge.

### 3.4. Survival risk prediction

A proposed survival tree can be used for prediction for a new sample not included in the training dataset. At each inner node, a positive  $Z$ -value ( $z_j^{d_0} > 0$ ) implies that the group of  $\{x_j > c_j\}$  has a higher risk of death than  $\{x_j \leq c_j\}$  does. Conversely, a negative  $Z$ -value ( $z_j^{d_0} < 0$ ) implies that the group of  $\{x_j \leq c_j\}$  has higher risk of death than  $\{x_j > c_j\}$  does. Thus, the signs of the  $Z$ -values in inner nodes determines the order of the death risks among the nodes.

We denote the ranks of the terminal nodes by  $R \in \{1, 2, \dots, k\}$ , where  $k$  is the number of terminal nodes in a tree. For instance, a terminal node has the highest risk  $R = k$  if it is judged 'higher risk' at all inner (ancestral) nodes. Similarly, the node of the lowest risk is

assigned for  $R = 1$ . To determine all other intermediate ranks, we impose a rule: any child node judged higher risk than the other child cannot be reversed by further splits.

The prediction for a new sample is performed by allocating the rank of the sample, and then calibrating the survival probability based on the samples in the corresponding node (e.g. by the Kaplan-Meier estimator). In the numerical analyses of this article, we define the left node to be the higher risk group and the right node to the lower risk group. In this way, all the terminal nodes are ordered by their risks from the highest risk (the leftmost node) to the lowest risk (the rightmost node). The tree becomes further informative by adding the Kaplan-Meier survival plot made by the samples in each terminal node.

The prediction performance of a tree can be assessed through the assigned ranks of risk. If test samples are available (in addition to training samples), one can compute the predicted risk ranks for all the test samples. Then, one can evaluate the concordance between the predicted risk ranks and the survival outcomes for all the samples in the test dataset. We adopt the  $c$ -index that is easily computed from censored data through the *concordance(.)* function in the *survival* package [40].

## 4. Simulations

Monte Carlo simulations were conducted to assess the performance of the proposed method and the existing methods (logrank tree and ctree) under high-dimensional covariates. The simulation consists of several steps: generating training data, constructing a tree, generating testing data, and evaluating the performance of the tree.

Since the evaluation of classification trees involves a few different aspects, we chose the following criteria:

- (i) *Selection ability*: We examine if the tree contains a reasonable proportion of informative nodes. The tree may have inner nodes that are falsely chosen (false positive). We use the ‘precision’ defined as the proportion of the inner nodes whose split uses informative covariates (true positive). A larger value of the precision corresponds to a better selection ability.
- (ii) *Classification ability*: We examine if the tree has a good classification ability. We use the likelihood ratio (LR) test for the equivalence of the classified groups based on the testing data. We use the  $P$ -value obtained from Cox regression treating terminal nodes as factors. A small  $P$ -value corresponds to a better classification ability.
- (iii) *Prognostic ability*: We examine if the tree has an ability to correctly assign the prognostic risk ranks for the testing data. We adopt the  $c$ -index that is easily computed from censored data through the *concordance(.)* function in the *survival* package.

### 4.1. Simulation designs

Data were simulated as follows. We generated  $p = 100$  discrete-valued gene expressions with

$$\mathbf{x}'_i = \left( \overbrace{(x_{i1}, \dots, x_{iq})}^{\times q}, \overbrace{(x_{iq+1}, \dots, x_{i2q})}^{\times q}, \overbrace{(x_{i2q+1}, \dots, x_{i100})}^{\times (100-2q)} \right),$$

$$\beta = (\overbrace{\beta, \dots, \beta}^{\times q}, \overbrace{-\beta, \dots, -\beta}^{\times q}, \overbrace{0, \dots, 0}^{\times(100-2q)}),$$

We considered a sparse setting ( $q = 10$ ) with  $\beta = 0.5$  or  $1$ , and a non-sparse setting ( $q = 30$ ) with  $\beta = 0.05$  or  $0.1$ . The intra-cluster correlation of gene expressions was  $0.8$  for the first two clusters of size  $q$ , and  $0$  for the last cluster of size  $100 - 2q$  (Emura *et al.* [17]). We first generated continuous gene expressions  $y_{ij} \sim Unif(-1.5, 1.5)$  from the R function `X.pathway(., rho1 = 0.8, rho2 = 0.8)` in the `compound.Cox` package. We then discretized them by the two scenarios:

**Balanced covariates:**

$$x_{ij} = \begin{cases} 1 & \text{if } -1.5 \leq y_{ij} < -0.75, \\ 2 & \text{if } -0.75 \leq y_{ij} < 0, \\ 3 & \text{if } 0 \leq y_{ij} < 0.75, \\ 4 & \text{if } -0.75 \leq y_{ij} \leq 1.5. \end{cases}$$

**Unbalanced covariates:**

$$x_{ij} = \begin{cases} 1 & \text{if } -1.5 \leq y_{ij} < -0.5, \\ 2 & \text{if } -0.5 \leq y_{ij} < 0.5, \\ 3 & \text{if } 0.5 \leq y_{ij} \leq 1.5. \end{cases} \tag{1}$$

For the latter, we randomly selected  $j \in \{2q + 1, \dots, 100\}$  and replace  $x_{ij}$  with  $4$ . Figure 3 is an example for  $q = 5$ , where  $j \in \{11, \dots, 100\}$  are randomly chosen for replacement.

Given the gene expressions, we generated  $T_i$  from a Cox model  $h(t|\mathbf{x}'_i) = h_0(t) \exp(\mathbf{x}'_i\beta)$  with  $h_0(u) = 1$ . We also generated  $U_i \sim Unif(0, 1)$ , and set  $t_i = \min\{T_i, U_i\}$  and  $\delta_i = I\{T_i \leq U_i\}$ . The censoring percentage is around  $50 \sim 56\%$ . Thus, we generated training data  $\{(t_i, \delta_i, \mathbf{x}_i); i = 1, \dots, n\}$  to develop trees under  $P$ -value thresholds of  $0.01$  and  $0.005$  (the conditional inference tree uses the adjusted thresholds (e.g. the adjusted value for  $0.01$  is  $1 - (1 - 0.01)^{100} \approx 0.633$ )). We also generated independent and identically distributed test samples in order to measure prediction accuracy. All simulations were based on  $100$  repetitions, where each repetition develops a tree and tests its prediction performance.

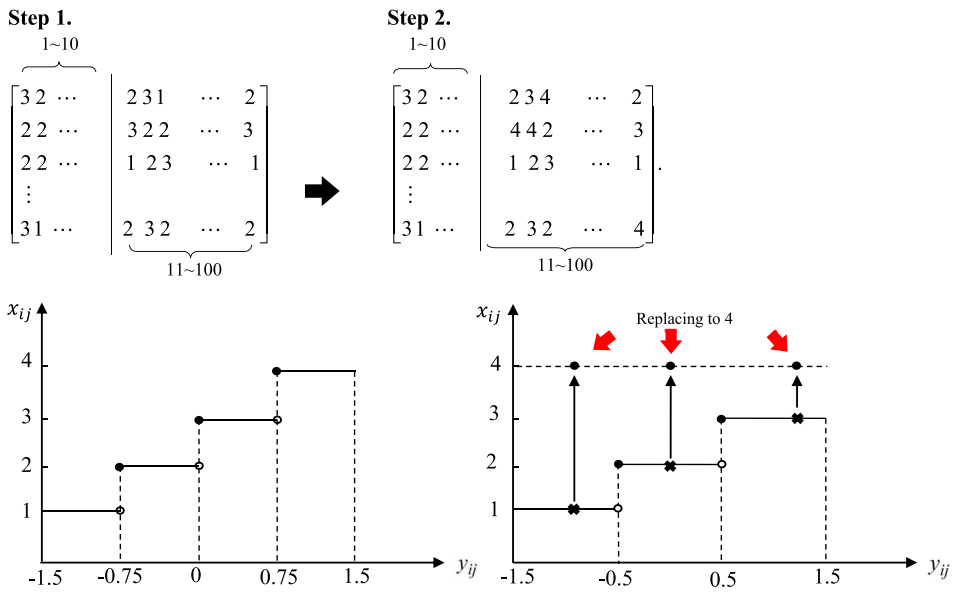
Supplementary Materials include additional stimulations where the data were generated from the continuous gene expressions (without discretization). We observe that the effect of discretization is minimal.

**4.2. Simulation result**

Tables 2 and 3 compare the performance of the proposed tree, the logrank tree, and the ctree.

If the covariates are balanced, the proposed tree and the logrank tree give almost identical performance in terms of the classification ability (LR test) and prognostic ability ( $c$ -index). While the ctree gives the lowest  $c$ -index, the classification ability is often the best. In fact, all the trees performed competitively well so that no clear-cut conclusion is derived to judge the best tree.

If the covariates are unbalanced, the performance of all the trees gets lower while the proposed tree exhibits a modest advantage over the logrank tree in terms of the LR test and the  $c$ -index. The improved results come from the fact that the proposed tree avoids false nodes by stabilizing the test results for unbalanced samples. This is exactly what we



**Figure 3.** Simulating unbalanced covariates (upper) and maps from  $x_{ij}$  to  $x'_{ij}$  (lower).

expected. However, the improvement under non-sparse setting is weaker than that under the sparse setting (Table 2 vs. Table 3). This means that the variance stabilization is effective specifically for the sparse setting.

We learn that the sizes of the tree (No. terminal nodes) are remarkably different among the three methods. In principle, a smaller tree is preferred to a larger tree if they have the same prediction performance. We observe that the logrank tree gives the largest tree while the ctree gives the smallest tree. In this respect, the ctree provides the most parsimonious model. However, the tree seems to be too small. Indeed, the low  $c$ -index for the ctree likely indicates the failure to have informative nodes in a tree. The proposed method gives an intermediate tree size.

Under the sparse setting (Table 2) the precision is usually less than 50% for all trees. Hence, the majority of the nodes are falsely selected. Nonetheless, the proposed tree has higher precision than the logrank tree. This improvement is due to the conservative tests results of the proposed tree relative to the logrank tree. The ctree should have the highest precision due to the smallest number of terminal nodes in a tree; however, we were unable to calculate the precision from the output.

Based on the above findings, we conclude that the proposed method effectively selects nodes from high-dimensional and unbalanced covariates. The proposed tree has a better prognostic ability than the ctree and a more precise selection ability than the logrank tree. However, under the balanced covariates, the performance of the all the trees are equally well.

### 5. Data analysis

This section applies the proposed method to the lung cancer data to illustrate how a survival tree is constructed, comparing the results with other tree-based methods and model-based methods.

**Table 2.** Simulation results comparing the four methods under the sparse setting (100 replications). The sample size is  $n = 100$  and the censoring percentage is around 50% ~ 56%. The score tests are stabilized by  $d_0 = 0.01$  and 0.1.

Split criterion: $P$ -value < 0.01			Logrank	Score 0.01	Score 0.1	ctree
$\beta = 1$	Balanced covariates	No. terminal nodes	18.48	18.03	13.05	3.66
		Precision%	42.8	44.7	54.3	–
		LR test: $\log_{10}(P)$	–11.14	–11.22	–12.03	–12.51
	Unbalanced covariates	$c$ -index	0.797	0.798	0.798	0.770
		No. terminal nodes	23.16	22.36	14.30	3.40
		Precision%	25.7	27.3	44.0	–
$\beta = 0.5$	Balanced covariates	LR test: $\log_{10}(P)$	–9.16	–9.50	–10.93	–9.22
		$c$ -index	0.759	0.765	0.784	0.742
		No. terminal nodes	17.93	17.58	12.51	3.18
	Unbalanced covariates	Precision%	37.0	38.7	46.4	–
		LR test: $\log_{10}(P)$	–7.87	–7.96	–8.42	–8.90
		$c$ -index	0.768	0.770	0.767	0.737
Split criterion: $P$ -value < 0.005	Balanced covariates	No. terminal nodes	21.56	21.11	13.92	2.93
		Precision%	21.6	23.0	35.1	–
		LR test: $\log_{10}(P)$	–5.52	–5.56	–6.96	–6.30
	Unbalanced covariates	$c$ -index	0.710	0.714	0.743	0.701
		No. terminal nodes	16.64	16.12	11.02	3.51
		Precision%	46.1	48.2	63.6	–
$\beta = 1$	Balanced covariates	LR test: $\log_{10}(P)$	–11.43	–11.69	–12.17	–12.26
		$c$ -index	0.796	0.797	0.796	0.765
		No. terminal nodes	21.50	20.21	12.03	3.16
	Unbalanced covariates	Precision%	26.7	29.0	50.9	–
		LR test: $\log_{10}(P)$	–9.24	–9.59	–11.17	–8.78
		$c$ -index	0.758	0.763	0.782	0.736
$\beta = 0.5$	Balanced covariates	No. terminal nodes	15.94	15.23	10.27	3.05
		Precision%	40.0	42.7	54.2	–
		LR test: $\log_{10}(P)$	–7.94	–8.16	–8.57	–8.51
	Unbalanced covariates	$c$ -index	0.768	0.769	0.765	0.731
		No. terminal nodes	19.78	19.23	11.42	2.71
		Precision%	22.5	24.1	40.9	–
Unbalanced covariates	LR test: $\log_{10}(P)$	–5.63	–5.60	–6.98	–5.91	
	$c$ -index	0.710	0.713	0.738	0.689	

Note: No. terminal nodes = the number of terminal nodes in a tree; Precision = the number of the true splits in a tree divided by the number of splits in a tree. LR test:  $\log_{10}(P)$  = the logarithmic value of the  $P$ -value of the LR test for testing the equality of the terminal nodes:  $\log_{10}(0.01) = -2.0$  and  $\log_{10}(0.001) = -3.0$ .

### 5.1. The lung cancer data

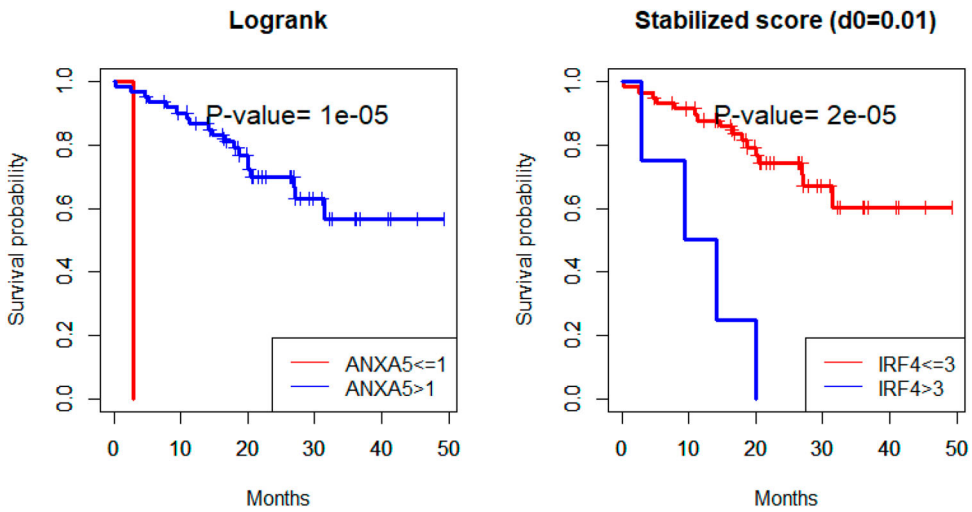
We consider the lung cancer data of Chen *et al.* [7] consisting of 125 lung cancer patients. Each patient has gene expressions from his/her tumor, coded as 1, 2, 3, or 4 (~ 25th, 25th ~ 50th, 50th ~ 75th, or 75th ~ percentile). The endpoint is overall survival (i.e. time-to-death due to any reason). During the follow-up, 38 patients died and other 87 patients were censored. In Chen *et al.* [7], the 125 patients were separated into 63 training and 62 testing samples. The difference between the training and testing sets was not found by significance tests [22].

**Table 3.** Simulation results comparing the four methods under the non-sparse setting (100 replications). The sample size is  $n = 100$  and the censoring percentage is around 50% ~ 56%. The score tests are stabilized by  $d_0 = 0.01$  and 0.1.

Split criterion: $P$ -value < 0.01			Logrank	Score 0.01	Score 0.1	ctree
$\beta = 0.1$	Balanced covariates	No. terminal nodes	17.47	16.93	12.21	3.52
		Precision%	0.751	0.766	0.799	–
		LR test: $\log_{10}(P)$	–9.48	–9.51	–9.82	–9.75
	Unbalanced covariates	c-index	0.786	0.786	0.786	0.751
		No. terminal nodes	18.60	17.95	11.98	3.47
		Precision%	0.559	0.573	0.686	–
$\beta = 0.05$	Balanced covariates	LR test: $\log_{10}(P)$	–7.79	–7.86	–8.01	–6.68
		c-index	0.749	0.749	0.752	0.713
		No. terminal nodes	16.95	16.44	11.32	2.81
	Unbalanced covariates	Precision%	0.667	0.685	0.702	–
		LR test: $\log_{10}(P)$	–5.30	–5.44	–5.42	–5.46
		c-index	0.725	0.726	0.723	0.682
Split criterion: $P$ -value < 0.005	Balanced covariates	No. terminal nodes	16.87	16.08	10.93	2.32
		Precision%	0.469	0.489	0.532	–
		LR test: $\log_{10}(P)$	–3.31	–3.28	–3.30	–2.640
	Unbalanced covariates	c-index	0.663	0.663	0.663	0.612
		No. terminal nodes	15.76	15.34	10.56	3.34
		Precision%	0.768	0.789	0.851	–
$\beta = 0.1$	Balanced covariates	LR test: $\log_{10}(P)$	–9.450	–9.520	–9.950	–9.490
		c-index	0.785	0.785	0.785	0.746
		No. terminal nodes	16.36	15.54	9.68	3.16
	Unbalanced covariates	Precision%	0.576	0.594	0.749	–
		LR test: $\log_{10}(P)$	–7.830	–7.730	–8.060	–6.320
		c-index	0.747	0.747	0.750	0.705
$\beta = 0.05$	Balanced covariates	No. terminal nodes	14.71	14.16	9.16	2.61
		Precision%	0.681	0.700	0.751	–
		LR test: $\log_{10}(P)$	–5.280	–5.370	–5.530	–5.200
	Unbalanced covariates	c-index	0.724	0.724	0.723	0.675
		No. terminal nodes	13.70	13.08	8.31	2.18
		Precision%	0.474	0.495	0.569	–
Unbalanced covariates	LR test: $\log_{10}(P)$	–3.220	–3.210	–3.230	–2.550	
	c-index	0.658	0.658	0.660	0.606	

Note: No. terminal nodes = the number of terminal nodes in a tree; Precision = the number of the true splits in a tree divided by the number of splits in a tree. LR test:  $\log_{10}(P)$  = the logarithmic value of the  $P$ -value of the LR test for testing the equality of the terminal nodes:  $\log_{10}(0.01) = -2.0$  and  $\log_{10}(0.001) = -3.0$ .

We use the subset of the data containing 97 gene expressions as available in the *Lung* object in the *compound.Cox* R package [19]. Available are survival time (time to either death or censoring) in months, censoring indicator (1 = death, or 0 = censoring), index for training sample (TRUE = training sample, or FALSE = testing sample), and genes named by their symbols, VHL, IHPK1, ..., RPL5. We shall construct survival trees based on the training samples ( $n = 63$ ), and validate their performance by the testing samples ( $n = 62$ ).



**Figure 4.** Kaplan-Meier estimates for the two separated groups by the logrank test (left panel) and stabilized score test (right panel) based on the training samples of  $n = 63$ . The  $P$ -value of the difference is computed by the logrank test.

**5.2. Results: the root node**

We illustrate the advantage of the stabilized score test over the logrank tree owing to unbalanced samples. Figure 4 shows how the logrank test optimally splits the  $n = 63$  samples into  $\{ANXA5 \leq 1\}$  vs.  $\{ANXA5 > 1\}$ .

In Figure 4, we see that the Kaplan-Meier plots for the two groups are not convincingly separated due to the unbalanced sample sizes. On the other hand, the stabilized score tests gave  $\{IRF4 \leq 3\}$  vs.  $\{IRF4 > 3\}$  as the best split. Indeed, the stabilized test gives a more convincing separation between the low and high risks (Figure 4).

**5.3. Results: survival trees**

From the  $n = 63$  training sample, survival trees were developed from the following methods:

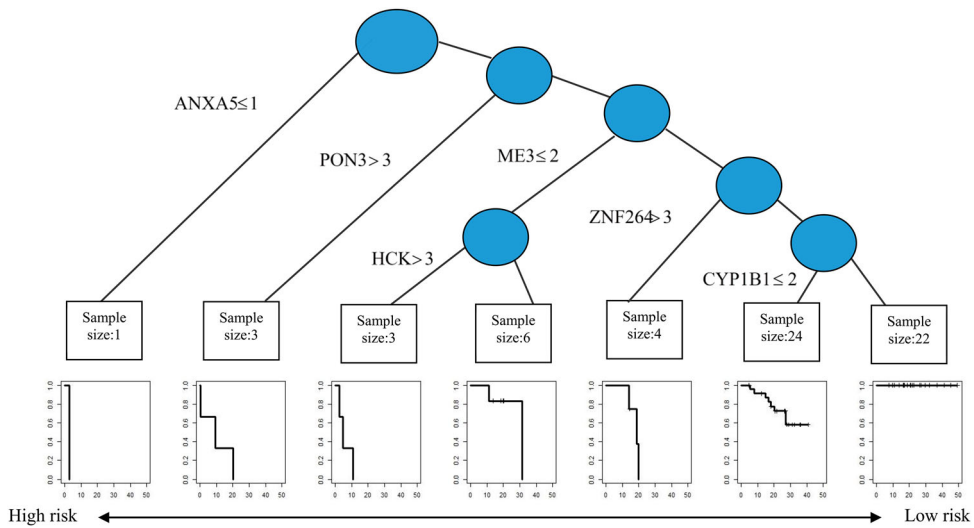
- (I) The logrank tree (using the score tests with  $d_0 = 0$ ),
- (II) The proposed tree (using the stabilized score tests with  $d_0 = 0.01$  chosen by AIC),
- (III) The conditional inference tree (ctree).

In (I) and (II), we set the stopping criterion of  $P = 0.01$ . Other choices of  $P$  were also examined (see Supplementary Materials for details). In (III), the choice of  $P$  shall be explained later.

- (I) The logrank tree

The root node splitted into  $\{ANXA5 \leq 1\}$  and  $\{ANXA5 > 1\}$  (Figure 5). Starting from the root node, the splitting process was recursively continued. The tree consisted of six inner nodes (including the root node) and seven terminal nodes.





**Figure 5.** The logrank tree constructed by the stopping criterion ( $P = 0.01$ ).

The Kaplan-Meier survival plots for the seven terminal nodes show that the prognosis groups are correctly ordered from the highest risk group (the leftmost node) to the lowest risk group (the rightmost group). One major concern is the too small sample size ( $n = 1$ ) for the highest risk group (Figure 5).

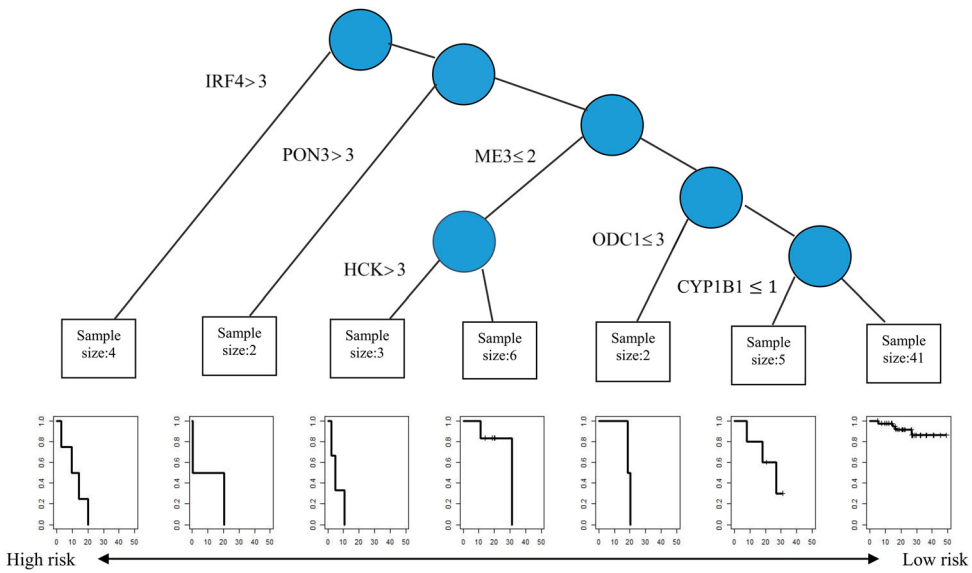
(II) The stabilized score (proposed) tree

The tree based on  $d_0 = 0.01$  (Figure 6) has the smallest AIC value among all the possible  $d_0 \geq 0$ . The root node was  $\{IRF4 \leq 3\}$  and  $\{IRF4 > 3\}$  (Figure 6). This split mitigates the major concern of the logrank tree. Except for the root node, the inner nodes are similar to those of the logrank tree.

The Kaplan-Meier survival plots for the six terminal nodes are consistent with the risk groups, correctly ordered from the highest risk group (the leftmost node) to the lowest risk group (the rightmost group).

(III) Conditional inference tree by the R function *ctree(.)*

The *ctree* under a threshold  $P = 0.01$  was performed (by *ctree(control = ctree\_control(alpha = 0.01, where alpha is the significance level for the splits)*). Consequently, the output returned the null tree consisting of only one terminal node of all samples since none of the splits reached the significance level of 0.01. However, this conclusion is not reasonable since the covariates should have some predictive ability for survival outcomes as previously reported [7,15,17,19]. To understand this phenomenon, one should note that the  $P$ -value threshold in *ctree(.)* is the Bonferroni adjusted value  $P^{Adj} = 1 - (1 - P)^p$ , where  $p$  is the number covariates and  $P$  is the non-adjusted  $P$ -value for individual tests (Hothorn *et al.* [26]). Since  $p = 97$  and  $P = 0.01$ , we set the adjusted  $P$ -value threshold  $P^{Adj} = 1 - (1 - 0.01)^{97} = 0.62$ . However, the resultant *ctree* still had



**Figure 6.** The stabilized score (proposed) tree with the shrinkage parameter  $d_0 = 0.01$  and the stopping criterion ( $P = 0.01$ ) based on the lung cancer data ( $n = 63$ ).

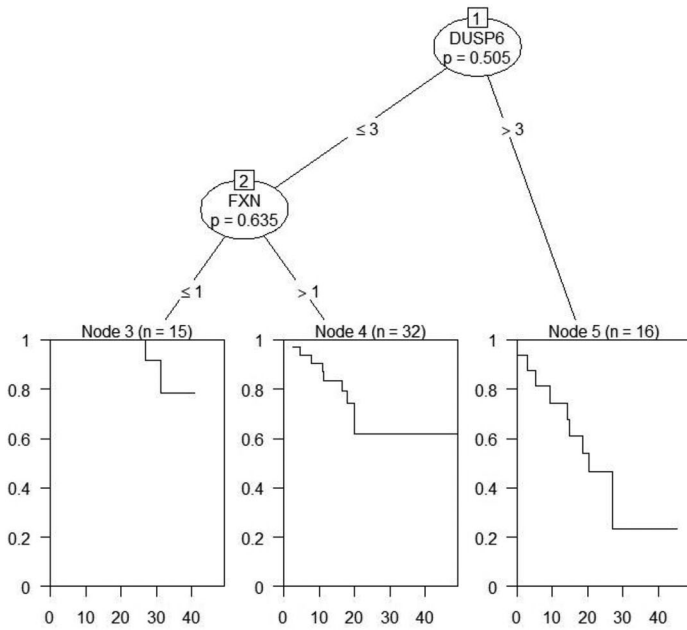
only two terminal nodes. Larger values of  $P^{Adj}$  yielded more reasonable trees with 3 terminal nodes ( $0.64 \leq P^{Adj} \leq 0.67$ , Figure 7a) and 4 terminal nodes ( $0.67 < P^{Adj} \leq 0.90$ , Figure 7b). See Supplementary Materials for our detailed process of choosing  $P^{Adj}$ .

Kaplan-Meier survival plots in Figure 7 do not exhibit a clear separation between the risk groups. However, it turns out that they have some prognostic ability when applied to the test samples. The split between  $\{DUSP \leq 3\}$  and  $\{DUSP > 3\}$  appears in the root node, which is the most significant split in the tree. However, the attached  $P$ -value to this split is ' $P = 0.505$ ', showing no evidence to separate the two groups in terms of survival prognosis. Thus, the input and output of the  $P$ -value thresholds are difficult to interpret, contradicting to the predictive performance. While the problem can be solved by adjusting the  $P$ -value, users may not know how to do it.

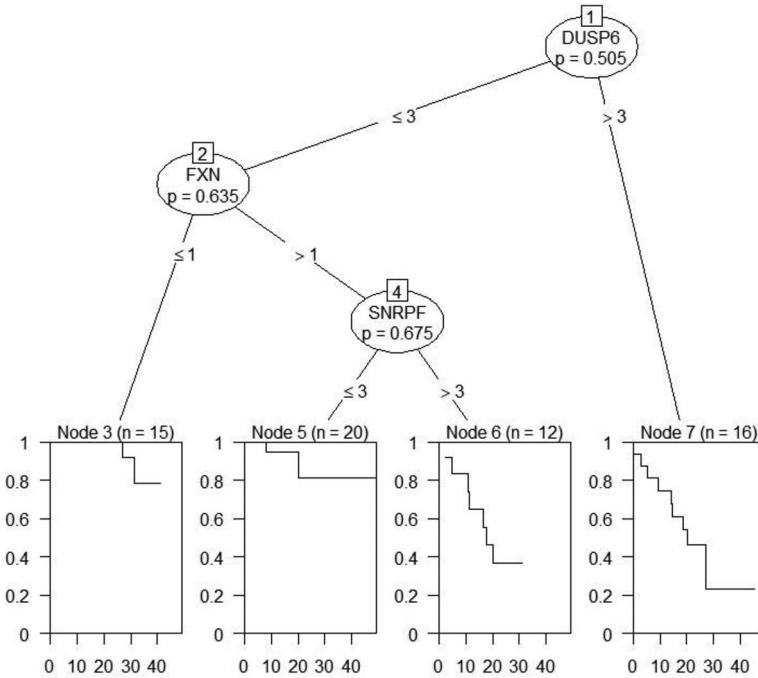
### 5.4. Validation results

We validated the prediction performance of the developed trees by the test samples of  $n = 62$ . For all the test samples, we predicted their risk groups using one of the trees. We then calculated the  $c$ -index, a concordance measure between the predicted risk ranks and the actual survival outcomes (Section 3.4). Higher values in the  $c$ -index means better prediction performance, and values less than 0.5 mean no prediction ability.

Table 4 shows the  $c$ -index for the five different trees. All the trees have similar values of the  $c$ -index ( $0.564 \sim 0.584$ ). With these small differences in the performance, it is not reasonable to select the best tree based on the largest  $c$ -index. The  $c$ -index values all not far from 0.5 are reasonable considering the small training samples ( $n = 63$ ), more than half of



(a) Stopping criterion:  $0.64 \leq P^{Adj} \leq 0.67$



(b) Stopping criterion:  $0.67 < P^{Adj} \leq 0.90$ .

**Figure 7.** The conditional inference tree (ctree) based on the lung cancer data ( $n = 63$ ). (a) Stopping criterion:  $0.64 \leq P^{Adj} \leq 0.67$  (b) Stopping criterion:  $0.67 < P^{Adj} \leq 0.90$ .

**Table 4.** Predictive performance (*c*-index) for the eleven methods based on the lung cancer data.

Method	Prediction formula	<i>c</i> -index
Logrank tree	See Figure 5	0.566
Score tree ( $d_0 = 0.01$ )	See Figure 6	0.564
Score tree ( $d_0 = 0.1$ )	Not shown	0.567
ctree ( $0.64 \leq \rho^{Adj} \leq 0.67$ )	See Figure 7(a)	0.569
ctree ( $0.67 < \rho^{Adj} \leq 0.90$ )	See Figure 7(b)	0.584
CC (Wald)	$(-1.09 \times ANXA5) + (1.32 \times DLG2) + (0.55 \times ZNF264) + (0.75 \times DUSP6) + (0.59 \times CPEB4) + (-0.84 \times LCK) + (-0.58 \times STAT1)$	0.560
CC (score)	$(-3.36 \times ANXA5) + (3.11 \times DLG2) + (2.81 \times ZNF264) + (2.71 \times DUSP6) + (2.53 \times CPEB4) + (-2.51 \times LCK) + (-2.45 \times STAT1) + (2.37 \times STAT2) + (2.35 \times RNF4) + (2.23 \times IRF4)$	0.570
Copula (Wald)	$(0.05 \times ANXA5) + (0.96 \times DLG2) + (0.53 \times ZNF264) + (0.41 \times DUSP6) + (0.42 \times CPEB4) + (-0.34 \times LCK) + (0.01 \times STAT1)$	0.580
Copula (score)	$(0.05 \times ANXA5) + (0.96 \times DLG2) + (0.53 \times ZNF264) + (0.41 \times DUSP6) + (0.42 \times CPEB4) + (-0.34 \times LCK) + (0.01 \times STAT1) + (0.43 \times STAT2) + (0.06 \times RNF4) + (0.30 \times IRF4)$	0.580
CS (Wald)	$(-0.58 \times ANXA5) + (0.95 \times DLG2) + (0.20 \times ZNF264) + (0.55 \times DUSP6) + (-0.25 \times CPEB4) + (-0.52 \times LCK) + (-0.23 \times STAT1)$	0.570
CS (score)	$(-0.61 \times ANXA5) + (0.89 \times DLG2) + (0.16 \times ZNF264) + (0.58 \times DUSP6) + (-0.36 \times CPEB4) + (-0.50 \times LK) + (-0.13 \times STAT1) + (0.47 \times STAT2) + (0.30 \times RNF4) + (-0.01 \times IRF4)$	0.600

Note: CC = compound covariate; Copula = copula method; CS = compound shrinkage; Wald = genes are selected by the Wald tests; score = genes are selected by the score tests. See [19] for details.

which were censored, and a number of possibly uninformative genes ( $p = 97$ ). However, we notice that the tree structures are different, especially between the proposed tree and the ctree. In fact, this phenomenon is typical if prediction formulas are derived from a large number of predictors. That is, there could be a number of prediction schemes all yielding the same performance in prediction.

Table 4 also shows the *c*-index for the six linear predictors as previously reported in Emura *et al.* [19]. Again, the *c*-index for these predictors are similar to those for the trees. However, we notice that trees' discrete predictors (ranking of 1, 2, ...) cannot be comparable to linear predictors' continuous predictors. Hence, this comparative *c*-index values may actually demonstrate the advantage of the trees over the continuous predictor. The continuous predictors usually need to be further splitted to yield clinically interpretable subgroups (e.g. good prognosis groups vs. poor prognosis group). Thus, we have validated all the trees by showing comparable predictive ability to the previously reported benchmark ability.

### 5.5. Conclusion of the data analysis

While we have validated all the trees (logrank tree, stabilized score tree, and ctree), the best one could not be selected by thier prediction performance. To discuss the best tree among the five trees, we search for the clinical interpretations for the trees. The proposed tree had an important gene 'IRF4' (Interferon Regulatory Factor) that is a strongly effective gene as a diagnostic and prognostic marker for human non-small cell lung cancer. Medical

researchers pointed out that the gene ‘IRF4’ is upregulated in this cancer (Alvisi *et al.* [1]), that is, a larger value of ‘IRF4’ leads to a higher risk for lung cancer patients. This evidence demonstrates the clinical advantage of the proposed tree over other trees. Besides, the gene ‘ANXA5’ selected by the logrank tree and all other existing predictors does not have a clear interpretation since it may not be a predictive gene of lung cancer. Instead, it is a prognostic marker of cervical cancer, and head and neck cancer (Kang *et al.* [29]).

In summary, we have demonstrated the advantage of the proposed method (the stabilized score tree) over the logrank tree, the ctrees, and other linear predictors in terms of the clinical relevance. However, all the methods exhibited similar numerical performance for prediction. Indeed, all the existing trees and prediction methods are highly sophisticated so that they all reached maximum levels in terms of quantitative prediction abilities.

## 6. Conclusions

We have proposed a new tree-based classification method motivated by some difficulties in the two most popular methods: the logrank tree and conditional inference tree. Motivated by the unusual detection of the survival difference by the logrank test, we stabilized the variance when computing the  $Z$ -value. This idea was previously proposed by Witten and Tibshirani [42] for testing a large number of covariates. The adaptation of their idea to a survival tree is a very reasonable approach, yet it has not been considered. We also have developed a computationally efficient method to find the optimal split among a large number of candidate splits. This extends the technique of Emura *et al.* [19].

Our simulations have demonstrated the improved precision and classification/prognostic ability of the proposed method over the existing methods. Specifically, the proposed method is advantageous when a number of unbalanced covariates tend to yield false positive nodes in the logrank tests. This is exactly the case where the stabilized score tests work better than the logrank tests. However, this advantage diminishes when the false positive rate is reduced in the presence of a large number of informative covariates (the non-sparse setting).

By applying the proposed tree to the lung cancer data, we have demonstrated the advantage of the proposed method (the stabilized score tree) over the logrank tree, the ctrees, and other continuous predictors in terms of the clinical relevance. However, all the methods exhibited similar performance for prediction.

The critical assumption made on survival data is that censoring mechanism is independent of survival time. The so-called ‘independent censoring assumption’ needs to be assumed in order to have valid results on the logrank test and score tests. In clinical survival data, some patients may drop out from the medical follow-up study due to their poor health, violating the independent censoring assumption [11,15,16]. Under such dependent censoring scenarios, the significance of the tests could be false. Thus, in a classification tree, the treatment of survival data with dependent censoring is an important issue as discussed by Moradian *et al.* [36]. They suggested a new measure of the binary splitting criterion by adjusting for dependence between survival time and censoring time by a copula-graphic estimator; see [18,32,33] for the applications of the copula-graphic estimator to estimate survival difference. Recently, Emura and Hsu [18] studied the performance of the two-sample test under copula-based dependent censoring models. A measure that

is free from the specification of a copula could also be considered under a competing risks framework [6]. A test based on the Fine-Gray model may also be considered by treating dependent censoring as a competing risk [2]. More explorations are necessary to deal with dependent censoring.

The so-called *survival forest* is a re-sampling version of a survival tree, which has gained popularity in recent years [25,28,36]. The extension of the proposed tree to develop a survival forest is a promising future work. While there could be several advantages for survival forests over survival trees, we believe that survival trees are easier to use in clinical practice due to its computational and conceptual simplicity.

The logrank test remains the most popular nonparametric method to detect the survival difference between two groups due to its simple computation and interpretation. The stabilized score test can be regarded as a minor correction to the logrank test to mitigate some unusual cases, keeping essentially the same computation, interpretation, and asymptotic distribution as the logrank test. One could achieve an improved tree for early or late survival difference from the null via weighted logrank tests [23]. The stabilization test is applicable to the weighted logrank statistics by simply correcting its variance estimate. This is a relevant generalization of the proposed stabilized tree especially when users focus on early or late survival benefit.

We are reluctant to use deviance-based measures that need a likelihood function under some parametric models. However, we note that deviance-based trees are common in continuous, binary, and count data, under the framework of the generalized linear model (GLM). Similarly, we have not considered entropy-based measures that needs a specific criterion, such as AIC, BIC, and Gini-index. For continuous (normally distributed) data, Yanagawa and Tajiri [43] suggested the AIC-based criterion as an alternative to the two-sample problem under small sample sizes. As for the correction to the small sample problems of the logrank tests in survival data, bootstrap and permutation methods are commonly used alternatives (Leblanc and Crowley [31]; Ditzhaus and Pauly [12]). There are growing interest in using the Mann–Whitney type effect for measuring survival difference under the independent censoring scenario [13,14,37] and the dependent censoring scenario [18,6]. These computationally expensive methods may provide some rooms for improvement.

## Acknowledgements

We thank Associate Editor and two anonymous reviewers for their helpful comments that greatly improved the manuscript. The research of Emura T is funded by the grant from the Ministry of Science and Technology of Taiwan (MOST 107-2118-M-008 -003-MY3).

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## Funding

The research of Emura T is funded by the grant from the Ministry of Science and Technology of Taiwan (MOST 107-2118-M-008 -003-MY3).

## References

- [1] G. Alvisi, J. Brummelman, S. Puccio, E.M. Mazza, E.P. Tomadam, A. Losurdo, V. Zanon, C. Peano, F.S. Colombo, A. Scarpa, M. Alloisio, A. Vasanthakumar, R. Roychoudhuri, M. Kallikourdis, M. Pagani, E. Lopci, P. Novellis, J. Blume, A. Kallies, G. Veronesi, and E. Lugli, *IRF4 instructs effector Treg differentiation and immune suppression in human cancer*. *J. Clin. Invest.* 130(6) (2020), pp. 3137–3150.
- [2] G. Bakoyannis, F.I. Chu, A.G. Babiker, and G. Touloumi, *Impact of covariate omission and categorization from the Fine–Gray model in randomized-controlled trials*. *Jpn. J. Stat. Data Sci.* (2021). doi:10.1007/s42081-021-00111-5.
- [3] D.G. Beer, S.L.R. Kardia, C.C. Huang, T.J. Giordano, A.M. Levin, D.E. Misek, L. Lin, G. Chen, T.G. Gharib, D.G. Thomas, M.L. Lizyness, R. Kuick, S. Hayasaka, J.M.G. Taylor, M.D. Iannettoni, M.B. Orringer, and S. Hanash, *Gene-expression profiles predict survival of patients with lung adenocarcinoma*. *Nat. Med.* 8 (2002), pp. 816–824.
- [4] I. Bou-Hamad, D. Larocque, and H. Ben-Ameur, *A review of survival trees*. *Stat. Surv.* 5 (2011), pp. 44–71.
- [5] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees*, Chapman and Hall, New York, 1984.
- [6] E. Cantagallo, M. De Backer, M. Kicinski, B. Ozenne, L. Collette, C. Legrand, M. Buyse, and J. Péron, *A new measure of treatment effect in clinical trials involving competing risks based on generalized pairwise comparisons*. *Biometrical J* 63 (2021), pp. 272–288. doi:10.1002/bimj.201900354.
- [7] H.Y. Chen, S.L. Yu, C.H. Chen, G.C. Chang, C.Y. Chen, A. Yuan, C.-L. Cheng, C.-H. Wang, H.-J. Terng, S.-F. Kao, W.-K. Chan, H.-N. Li, C.-C. Liu, S. Singh, W.J. Chen, J.J.W. Chen, and P.-C. Yang, *A five-gene signature and clinical outcome in non-small-cell lung cancer*. *N Engl. J. Med.* 356 (2007), pp. 11–20.
- [8] J. Choi, I. Oh, S. Seo, and J. Ahn, *G2vec: Distributed gene representations for identification of cancer prognostic genes*. *Sci. Rep* 8(1) (2018), pp. 1–10.
- [9] A. Ciampi, R.S. Bush, M. Gospodarowicz, and J.E. Till, *An approach to classifying prognostic factors related to survival experience for non-Hodgkin’s lymphoma patients: Based on a series of 982 patients: 1967–1975*. *Cancer* 47 (1981), pp. 621–627.
- [10] A. Ciampi, J. Thiffault, J.-P. Nakache, and B. Asselain, *Stratification by stepwise regression, correspondence analysis and recursive partition: a comparison of three methods of analysis for survival data with covariates*. *Comput. Stat. Data. Anal.* 4(3) (1986), pp. 185–204.
- [11] N.W. Deresa, and I. Van Keilegom, *A multivariate normal regression model for survival data subject to different types of dependent censoring*. *Comput. Stat. Data. Anal.* 144 (2020), p. 106879.
- [12] M. Ditzhaus, and M. Pauly, *Wild bootstrap logrank tests with broader power functions for testing superiority*. *Comput. Stat. Data. Anal.* 136 (2019), pp. 1–11.
- [13] D. Dobler, and M. Pauly, *Bootstrap- and permutation-based inference for the Mann–Whitney effect for right-censored and tied data*. *Test* 27(3) (2018), pp. 639–658.
- [14] D. Dobler, and M. Pauly, *Factorial analyses of treatment effects under independent right-censoring*. *Stat. Methods Med. Res.* 29(2) (2020), pp. 325–343.
- [15] T. Emura, and Y.H. Chen, *Gene selection for survival data under dependent censoring: A copula-based approach*. *Stat. Methods Med. Res.* 25(6) (2016), pp. 2840–2857.
- [16] T. Emura, and Y.H. Chen, *Analysis of Survival Data with Dependent Censoring, Copula-Based Approaches*, JSS Research Series in Statistics, Springer, Singapore, 2018.
- [17] T. Emura, Y.H. Chen, and H.Y. Chen, *Survival prediction based on compound covariate under Cox proportional hazard models*. *PLoS ONE* 7 (2012), p. e47627. doi:10.1371/journal.pone.0047627.
- [18] T. Emura, and J.H. Hsu, *Estimation of the Mann-Whitney effect in the two-sample problem under dependent censoring*. *Comput Stat Data Anal* 150 (2020), p. 106990.
- [19] T. Emura, S. Matsui, and H.Y. Chen, *Compound.Cox: Univariate feature selection and compound covariate for predicting survival*. *Comput. Methods Programs Biomed.* 168 (2019), pp. 21–37.

- [20] T. Emura, M. Nakatochi, S. Matsui, H. Michimae, and V. Rondeau, *Personalized dynamic prediction of death according to tumour progression and high-dimensional genetic factors: Meta-analysis with a joint model*. *Stat. Methods Med. Res.* 27(9) (2018), pp. 2842–2858.
- [21] B.S. Everitt, and D.C. Howell, *Classification and Regression Trees, Encyclopedia of Statistics in Behavioral Science*, 2nd ed., Wiley, Chichester, 2005, pp. 287–290.
- [22] L. Feng, X. Zhang, and B. Liu, *A high-dimensional spatial rank test for two-sample location problems*. *Comput. Stat. Data. Anal.* 144 (2020), p. 106889.
- [23] T.R. Fleming, and D.P. Harrington, *Counting Processes and Survival Analysis*, John Wiley & Sons, New York, 1991.
- [24] T. Hothorn, and B.S. Everitt, *A Handbook of Statistical Analyses Using R*, 3rd ed., CRC press, Boca Raton, 2014.
- [25] T. Hothorn, K. Hornik, and A. Zeileis, *Unbiased recursive partitioning: A conditional inference framework*. *J. Comput. Graph. Stat.* 15 (2006), pp. 651–674.
- [26] T. Hothorn, K. Hornik, and A. Zeileis, *ctree: Conditional Inference Trees*. CRAN Version 1.2–8 (2020).
- [27] T. Hothorn, H. Seibold, and A. Zeileis, *partykit: A toolkit for recursive partytioning*. CRAN Version 1.2-8 (2020).
- [28] H. Ishwaran, U.B. Kogalur, E.H. Blackstone, and M.S. Lauer, *Random survival forests*. *Ann. Appl. Stat.* 2(3) (2008), pp. 841–860.
- [29] T.H. Kang, J.H. Park, A. Yang, H.J. Park, S.E. Lee, Y.S. Kim, G.-Y. Jang, E. Farmer, B. Lam, Y.-M. Park, and C.-F. Hung, *Annexin A5 as an immune checkpoint inhibitor and tumor-homing molecule for cancer treatment*. *Nat. Commun* 11 (2020), pp. 1137.
- [30] M. Kim, I. Oh, and J. Ahn, *An improved method for prediction of cancer prognosis by network learning*. *Genes. (Basel)* 9(10) (2018), pp. 478.
- [31] M. LeBlanc, and J. Crowley, *A review of tree-based prognostic models*. *Cancer Res. Treat* 75 (1995), pp. 113–124.
- [32] S.M. Lo, E. Mammen, and R.A. Wilke, *A nested copula duration model for competing risks with multiple spells*. *Compt. Stat. Data Anal.* 150 (2020), p. 106986.
- [33] S.M. Lo, and R.A. Wilke, *A copula model for dependent competing risks*. *J. R. Stat. Soc. C Appl. Stat.* 59(2) (2010), pp. 359–376.
- [34] S. Matsui, *Predicting survival outcomes using subsets of significant genes in prognostic marker studies with microarrays*. *BMC Bioinform* 7(1) (2006), p. 156.
- [35] S. Matsui, *Statistical Issues in Clinical Development and Validation of Genomic Signatures, Design and Analysis of Clinical Trials for Predictive Medicine*, CRC Press, Boca Raton, 2015, pp. 207–226.
- [36] H. Moradian, D. Larocque, and F. Bellavance, *Survival forests for data with dependent censoring*. *Stat. Methods Med. Res.* 28(2) (2019), pp. 445–461.
- [37] J. Péron, M. Buyse, B. Ozenne, L. Roche, and P. Roy, *An extension of generalized pairwise comparisons for prioritized outcomes in the presence of censoring*. *Stat. Methods Med. Res.* 27 (2018), pp. 1230–1239.
- [38] M. Radespiel-Tröger, T. Rabenstein, H.T. Schneider, and B. Lausen, *Comparison of tree-based methods for prognostic stratification of survival data*. *Artif. Intell. Med.* 28 (2003), pp. 323–341.
- [39] T.M. Therneau, and E.J. Atkinson, *rpart: Recursive partitioning and regression trees*. CRAN Version 4.1–15 (2019).
- [40] T.M. Therneau, and T. Lumley, *survival: Survival analysis*. CRAN Version 3.1–12 (2020).
- [41] W.N. van Wieringen, D. Kun, R. Hampel, and L. Boulesteix, *Survival prediction using gene expression data: A review and comparison*. *Comput. Stat. Data Anal.* 53(5) (2009), pp. 1590–1603.
- [42] D.M. Witten, and R. Tibshirani, *Survival analysis with high-dimensional covariates*. *Stat. Methods Med. Res.* 19 (2010), pp. 29–51.
- [43] T. Yanagawa, and R. Tajiri, *Usefulness of Akaike information criterion for making decision in two-sample problems when sample sizes are too small*. *Jpn. J. Stat. Data Sci.* 1(2) (2018), pp. 333–346.



- [44] Q. Zhang, J. Wang, M. Liu, Q. Zhu, Q. Li, C. Xie, C. Han, Y. Wang, M. Gao, and J. Liu, *Weighted correlation gene network analysis reveals a new stemness index-related survival model for prognostic prediction in hepatocellular carcinoma*. *Aging* 12(13) (2020), pp. 13502–13517.

## Appendix. Computer software

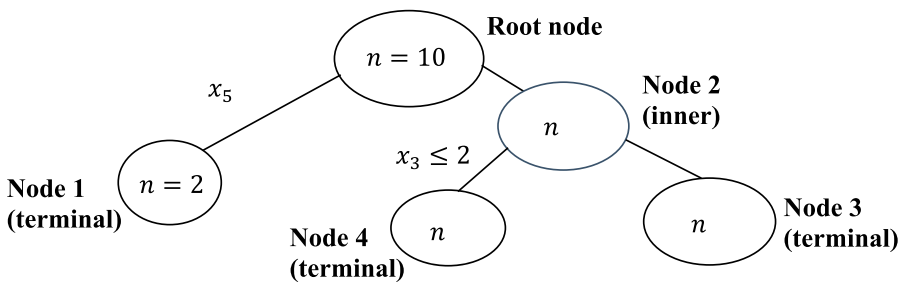
We made our original R package ‘*uni.survival.tree*’ to implement the proposed methods. Presently, there are seven functions in this package. Due to the space limitation, we focus on the most important function ‘*uni.tree(.)*’ that produces a survival tree given a survival dataset  $\{(t_i, \delta_i, \mathbf{x}_i); i = 1, 2, \dots, n\}$ . We also explain the function ‘*risk.classification(.)*’ that is useful for predicting the risk of death given a tree made by *uni.tree(.)* and test samples. We also briefly explain two functions for generating data, which was used in the simulations.

We use the following styles for inputting the dataset (similar to the *compound.Cox* R package),

- t.vec: a vector  $(t_1, t_2, \dots, t_n)$ ,
- d.vec: a vector  $(\delta_1, \delta_2, \dots, \delta_n)$ ,
- X.mat: a matrix with the  $i$ -th row  $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})$  for  $i = 1, \dots, n$ ,
- P.value: the  $P$ -value threshold ( $0 < P < 1$ ),
- d0: the shrinkage parameter  $d_0 > 0$  that stabilizes the score test,
- score: = TRUE for the score test; = FALSE for the logrank test.

The dataset of  $n = 10$  in Example 3.1 is used for illustration under  $P = 0.05$ . The code below creates t.vec, d.vec, and X.mat, and apply them to the *uni.tree(.)* function.

The output shows all the nodes created. The output ‘node\_status:’ tells us whether the node is ‘inner’ or ‘terminal’. First, we identify two inner nodes in the above output, corresponding to the root node and Node 2 in Figure 8. Their  $P$ -values are less than 0.05, indicating that these nodes have their child nodes. Next, we identify three terminal nodes in the above output, corresponding to Nodes 1, 3, and 4 in Figure 8. The  $P$ -values for the terminal nodes are all greater than 0.05, indicating that these nodes have no child. The output ‘Risk score’ can be used to assign the rank of death risk. The largest value ‘1’ corresponds to the highest risk group (Node 1, Rank = 3) and the smallest value ‘-1.1’ corresponds to the lowest risk group (Node 3, Rank = 1).



**Figure 8.** A survival tree constructed by using the data of Example 3.1.

```

library(uni.survival.tree)

t.vec=1:10
d.vec=c(0,0,1,0,1,0,1,1,0,1)
X.mat=cbind(
  x1=c(2,1,2,3,4,3,4,4,1,1),
  x2=c(1,3,3,2,2,1,1,2,4,4),
  x3=c(2,2,3,3,3,2,4,2,1,1),
  x4=c(4,3,1,2,3,4,1,2,3,2),
  x5=c(4,1,1,4,4,3,3,2,2,4)
)
res=uni.tree(t.vec,d.vec,X.mat,P.value=0.05,d0=0.01,score=TRUE)
res

$NODE
      Information
node_status: Inner.node
Risk score: 0
P-value: 0.0102257862304616
name: x5
cut_value: 1
Z-value -2.57

$Left
$Left$NODE
      Information
node_status: terminal node
Risk score: 1
P-value: 1
samplesize: 2
condition: x5<=1

$Right
$Right$NODE
      Information
node status: Inner.node
Risk score: -1
P-value: 0.0293192585110154
name: x3
cut_value: 2
Z-value 2.18

$Right$Left
$Right$Left$NODE
      Information
node_status: terminal node
Risk score: -0.9
P-value: 0.326892579781518
samplesize: 3
condition: x5>1 & x3>2

$Right$Right
$Right$Right$NODE
      Information
node_status: terminal node
Risk score: -1.1
P-value: 0.166102394264582
samplesize: 5
condition: x5>1 & x3<=2

```

To predict the risk of death for test samples, one has to identify the group that the samples belong (Section 3.4). The test samples are those who do not belong to the dataset. Nonetheless, we illustrate the predicted risk of death for all the same patients in the training dataset of  $n = 10$ . Inputting the

'x.mat' to the function *risk.classification(.)*, we identify the ranks (1 = low risk; 2 = medium risk; 3 = high risk) for 10 patients as follows:

```
risk.classification(res,X.mat)
  1  2  3  4  5  6  7  8  9 10
  1  3  3  2  2  1  2  1  1  1
```

For example, the first sample has the rank 1 (low risk). The low risk group has five samples (1, 6, 8, 9, and 10). With the ranks assigned to all the patients, one can compute the *c*-index between the predictor (ranks) and outcomes using the *concordance(.)* function. The *c*-index evaluates the prediction performance of a tree. Testing samples have to be prepared to calculate the *c*-index.

To generate the gene expressions used in the simulation designs, we made our original R functions *X.pathway\_discrete.balanced(.)* and *X.pathway\_discrete.imbalanced(.)* in the *uni.survival.tree* package. These functions are variants of *X.pathway(.)* function from the *compound.Cox* package (Emura *et al.* [17,19]). We show some example made by these two function. One needs to input the intra-cluster correlation 0.8 for the first two clusters and others ( $n = 10, p = 15, q_1 = 5, q_2 = 5$ ) into *X.pathway\_discrete.balanced(.)*. The output is discrete valued:

```
> X.pathway_discrete.balanced(10,15,5,5,rho1=0.8,rho2=0.8)
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]
[1,]  4   4   4   3   4   4   4   4   4   4   2   2   4   1   1
[2,]  4   4   4   4   4   4   4   4   4   4   4   1   2   1   1
[3,]  4   3   3   3   4   3   2   4   4   3   1   3   2   1   2
[4,]  4   3   3   3   3   4   4   3   4   3   1   2   1   1   3
[5,]  4   4   3   4   4   3   4   4   4   4   1   2   1   4   3
[6,]  4   4   4   4   4   4   4   4   4   4   1   1   1   2   2
[7,]  4   4   4   4   2   4   4   4   4   4   1   3   1   3   2
[8,]  4   4   4   4   4   3   4   4   4   4   1   3   3   1   2
[9,]  4   4   4   4   4   3   2   1   2   4   2   2   3   2   3
[10,] 4   4   4   4   4   4   3   4   4   4   1   4   1   3   4
```

*X.pathway\_discrete.imbalanced(.)* shows the imbalanced covariates for the last five columns:

```
> X.pathway_discrete.imbalanced(10,15,5,5,rho1=0.8,rho2=0.8)
  [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]
[1,]  1   2   2   2   2   2   2   2   3   2   4   1   4   2   3
[2,]  2   1   1   1   2   3   2   2   1   2   1   3   2   4   2
[3,]  2   2   2   3   1   1   2   1   1   1   3   2   2   3   1
[4,]  1   1   1   1   1   2   2   2   2   3   1   3   2   2   3
[5,]  1   2   2   1   2   3   2   3   2   3   2   1   3   1   2
[6,]  1   1   1   1   1   1   2   2   1   1   3   3   1   1   4
[7,]  3   3   3   3   2   2   1   2   2   2   1   1   1   2   3
[8,]  2   2   3   2   3   2   3   3   3   2   2   3   3   2   2
[9,]  1   1   1   2   1   2   1   2   2   2   2   3   3   3   2
[10,] 2   2   2   3   2   2   1   2   2   1   3   4   3   2   1
```