R package PRIMsrc: Bump Hunting by Patient Rule Induction Method for Survival, Regression and Classification

Jean-Eudes Dazard [*] Michael Choe [†] Michael LeBlanc [‡] J. Sunil Rao [§]

Abstract

PRIMsrc is a novel implementation of a non-parametric bump hunting procedure, based on the Patient Rule Induction Method (PRIM), offering a unified treatment of outcome variables, including censored time-to-event (Survival), continuous (Regression) and discrete (Classification) responses. To fit the model, it uses a recursive peeling procedure with specific peeling criteria and stopping rules depending on the response. To validate the model, it provides an objective function based on prediction-error or other specific statistic, as well as two alternative cross-validation techniques, adapted to the task of decision-rule making and estimation in the three types of settings. PRIMsrc comes as an open source R package, including at this point: (i) a main function for fitting a Survival Bump Hunting model with various options allowing cross-validated model selection to control model size (#covariates) and model complexity (#peeling steps) and generation of cross-validated end-point estimates; (ii) parallel computing; (iii) various S3-generic and specific plotting functions for data visualization, diagnostic, prediction, summary and display of results. It is available on CRAN and GitHub.

Key Words: Bump Hunting, Non-Parametric Methods, Rule-Induction Methods, Cross-Validation, Parallel Programming, R Package

1. Introduction

Non-Parametric Methods for Bump Hunting

The search for structures in datasets in the form of bumps, modes, components, clusters or classes are important as they often reveal underlying phenomena leading to scientific discoveries. Exploratory bump hunting seeks bump supports (possibly disjoint regions) of the input space of multi variables where a target function (e.g. a regression or density function) is on average larger (or lower) than it's average over the entire input space. Exploratory bump hunting covers tasks such as: (i) Mode(s) Hunting, (ii) Local/Global Extremum(a) Finding, (iii) Subgroup(s) Identification, (iv) Outlier(s) Detection.

Supervised bump hunting procedures are among the few non-parametric methods that have been proposed to address this problem. One known as the Patient Rule Induction Method (PRIM) was initially introduced by Friedman & Fisher [15] and later formalized by Polonik [23]. Essentially, the method is a recursive peeling algorithm that explores the input space solution region, where the response is expected to be larger on average. Some interesting features common and distinct to decision trees such as Classification and Regression Trees (CART) [3] help describe PRIM. For instance, often times, interest focuses only on estimating *extreme* response groups. In this respect, bump hunting is a better approach than decision trees because it aims at searching regions where the response function is larger (or smaller) than its average over the entire space. Other basic differences between decision-tree and decision-box methods lie in their approach and goal (reviewed in [5]).

To date, only a few extensions of the original PRIM work have been done. This includes a Bayesian model-assisted formulation of PRIM [29], a boosted version of PRIM based on Adaboost [28], an extension of PRIM to censored responses [19, 20] and to discrete variables [18]. Although PRIM is intrin-

[*]Division of Bioinformatics, Center for Proteomics and Bioinformatics, Case Western Reserve University. Cleveland, OH 44106, USA. Corresponding author Email (JED): jxd101@case.edu

[†]Division of Bioinformatics, Center for Proteomics and Bioinformatics, Case Western Reserve University. Cleveland, OH 44106, USA.

[‡]Department of Biostatistics, School of Public Health, University of Washington, Seattle, WA 98195, USA; Public Health Sciences, Fred Hutchinson Cancer Research Center, Seattle, WA 98109.

[§]Division of Biostatistics, Dept. of Epidemiology and Public Health, The University of Miami. Miami, FL 33136, USA.

sically multivariate, it was uncertain from the original work how the algorithm would perform in ultra high-dimension where collinearity [14, 15] and sparsity abound. So, recently, an interesting body of work studied when and why the Principal Component space can be used effectively to optimize the response-predictor relationship in bump hunting. This was first addressed in [8], where the computational details of such an approach were laid out for high-dimensional settings, then in [9, 10], where authors showed how the PC rotation of the predictor space alone can generate "improved" bump estimates.

Package Overview
Our `PRIMsrc` package is an implementation in the R-language of procedures derived from the Patient Rule Induction Method (PRIM) for fitting and validating a bump hunting model. `PRIMsrc` is intended to offer a unified treatment of various types of outcome variables including censored time-to-event (Survival), continuous (Regression) and discrete (Classification) responses. To fit the model, `PRIMsrc` uses a recursive peeling procedure with specific peeling criteria and stopping rules depending on the response.

One of the critiques made in the original PRIM work was the lack of validation procedure and measures of significance of solution regions. To optimize model parameters tuning and validate the model, `PRIMsrc` provides an objective function based on prediction-error or other specific statistic depending on the response, as well as a resampling technique amenable to the joint task of decision rule making by recursive peeling (i.e. decision-box) and estimation in the three types of settings. Specifically, `PRIMsrc` offers two alternative, possibly repeated, K-fold cross-validation techniques adapted to the task, namely the "Replicated Combined CV" (RCCV) and "Replicated Averaged CV" (RACV).

So far, the current version (0.6.0) is a development release that only implements the case of a survival response. Other features will be added soon. To build our survival/risk bump hunting model, `PRIMsrc` implements our "Patient Recursive Survival Peeling" (PRSP) method, a non-parametric recursive peeling procedure, derived from the Patient Rule Induction Method (PRIM), which we have extended to allow for survival/risk response, possibly censored. Specifically, `PRIMsrc` offers three appropriate peeling criteria derived from non/semi-parametric survival statistics such as the hazards-ratio, the log-rank test or the Nelson–Aalen estimator, that can be used alternatively to fit our survival/risk bump hunting model.

`PRIMsrc` is also intended to handle continuous and discrete input variables in low- and high-dimensional settings, including the situation where the number of variables exceeds that of samples ($p > n$ or $p \gg n$ paradigm). As such, the package includes cross-validation procedures to control model size (# covariates) in addition to model complexity (# peeling steps). It has been tested in multiple (> 20) low and high-dimensional situations where $n \leqslant p$ and even $n \ll p$ (see abstract of application article [6] and the example datasets in our R package).

2. Bump Hunting Framework

2.1 Notation - Goal

The formal setup of bump hunting is as follows (see also [5, 15, 23]). Let us consider a supervised problem with a univariate output (response) random variable, denoted $\mathbf{y} \in \mathbb{R}$. Further, let us consider a p-dimensional random vector $\mathbf{X} \in \mathbb{R}^p$ of support S, also called input space, in an Euclidean space. Let us denote the p input variables by $\mathbf{X} = [\mathbf{x}_j]_{j=1}^{p}$, of joint probability density function $p(\mathbf{X})$ and by $f(\mathbf{x}) = E(\mathbf{y}|\mathbf{X} = \mathbf{x})$ the target function to be optimized (e.g. any regression function or e.g. the p.m.f or p.d.f $f_{\mathbf{X}}(\mathbf{x})$).

Briefly, the goal in bump hunting is to find a sub-space or region ($R \subseteq S$) of the input space within which the average value $\bar{f}_R$ of $f(\mathbf{x})$ is expected to be significantly larger (or smaller) than its average value $\bar{f}_S$ over the entire input space S (see figures in [5]). In addition, one wishes that the corresponding support (mass) of R, say β_R, be not too small, that is, greater than a minimal support threshold, say $0 < \beta_0 < 1$.

Formally, in the continuous case of $\mathbf{X}$:

$$\bar{f}_R = \frac{\int_{\mathbf{x} \in R} f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}}{\int_{\mathbf{x} \in R} p(\mathbf{x}) d\mathbf{x}} \gg \bar{f}_S \qquad \beta_R = \int_{\mathbf{x} \in R} p(\mathbf{x}) d\mathbf{x} \gg \beta_0$$

In supervised problems with an output variable (response) $\mathbf{y}$, one would seek to characterize the conditional expectation $E(\mathbf{y}|\mathbf{X} = \mathbf{x})$ and infer the properties of the unknown joint probability density function $p(\mathbf{X})$, whereas in the case of unsupervised learning, one would have to directly infer the properties of $p(\mathbf{X})$, e.g. from some density estimate, without the help of a response.

Let S_j be the support of the jth variable $\mathbf{x}_j$, such that the input space can be written as the (Cartesian) outer product space $S = \times_{j=1}^{p} S_j$. Let $s_j \subseteq S_j$ denotes the unknown subset of values of variable $\mathbf{x}_j$ corresponding to the unknown support of the solution region R. Let $J \subseteq \{1, \ldots, p\}$ be the subset of indices of selected variables in the process. The goal in bump hunting amounts to finding the value-subsets $\{s_j\}_{j \in J}$ of the corresponding variables $\{\mathbf{x}_j\}_{j \in J}$ such that

$$R = \left\{ \mathbf{x} \in \bigcap_{j \in J} (\mathbf{x}_j \in s_j) : (\bar{f}_R \gg \bar{f}_S)(\beta_R \gg \beta_0) \right\}$$

2.2 Estimates

Since the underlying distribution is not known, the estimates of $\bar{f}_R$ and β_R must be used. Assume a supervised setting, where the outcome response variable is $\mathbf{y} = (y_1 \ldots y_n)^T$ and the explanatory/input variables are $\mathbf{X} = (\mathbf{x}_1 \ldots \mathbf{x}_n)^T$, where each observation is the p-dimensional vector of covariates $\mathbf{x}_i = [x_{i,1} \ldots x_{i,p}]^T$, for $i \in \{1, \ldots, n\}$. Plug-in estimates of the average value $\bar{f}_R$ of the target function $f(\mathbf{x})$ and of the support β_R of the region R are respectively derived as:

$$\hat{\bar{f}}_R = \frac{1}{n\hat{\beta}_R} \sum_{\mathbf{x}_i \in \hat{R}} y_i = \frac{1}{n\hat{\beta}_R} \sum_{i=1}^{n} y_i I(\mathbf{x}_i \in \hat{R}) \qquad \hat{\beta}_R = \frac{1}{n} \sum_{\mathbf{x}_i \in \hat{R}} I(\mathbf{x}_i \in \hat{R}) = \frac{1}{n} \sum_{i=1}^{n} I(\mathbf{x}_i \in \hat{R})$$

2.3 Remarks

1. The goal amounts to comparing the conditional expectation of the response over the solution region R: $\bar{f}_R = E[f(\mathbf{x})|\mathbf{x} \in R]$ with the unconditional one $\bar{f}_S = E[f(\mathbf{x})]$.
2. Larger target function average $\bar{f}_R$ is associated with smaller support β_R of the region R (Figure 1). So, in practice, there is a trade-off between maximizing $\bar{f}_R$ and maximizing β_R.
3. If the target function to be optimized is for instance the p.m.f or p.d.f $f_{\mathbf{X}}(\mathbf{x})$, then $\Pr(\mathbf{X} \in R)$ is the probability mass/density of a local maximum and the task is equivalent to a mode(s) hunting.
4. In the case of real-valued inputs, the entire input space is the p-dimensional outer product space $S \subseteq \mathbb{R}^p$; the support S_j of each individual input variable (and of each corresponding value-subset s_j) is the usual interval of the form $S_j = \left[t_j^-, t_j^+\right] \subset \mathbb{R}$ for $j = 1, \ldots, p$; the solution region R has the shape of a $|J|$-dimensional hyper-rectangle in $\mathbb{R}^{|J|}$, called a *box*, which can be written as the outer product of $|J|$ intervals of the form $B = \times_{j \in J} [t_j^-, t_j^+]$.
5. In general, region R could be any smooth shape (e.g. a convex hull) possibly disjoint.

2.4 Estimation by the Patient Rule Induction Method (PRIM)

The Patient Rule Induction Method (PRIM) is used to get the region estimate $\hat{R}$ with corresponding support estimate $\hat{\beta}_R$ and conditional output response mean estimate $\hat{\bar{f}}_R$. Essentially, the method is one of recursive peeling/pasting algorithm (a discrete version of the steepest ascent method) that explores the input space solution region, where the response is expected to be larger on average. The method generates a sequence of boxes that collectively cover the region estimate $\hat{R}$. The way the box peeling/induction is done and the space is covered, as well as how the patience is controlled and the stopping rule is used are detailed in the companion article (see [5]) and the original article of Friedman & Fisher [15], later formalized by Polonik & Wang [23].

3. Survival Bump Hunting by Recursive Peeling

Assume a univariate survival/risk response variable (possibly censored) in a multivariate setting of real-valued (continuous or discrete) input variables/covariates $\mathbf{X} = [\mathbf{x}_j]_{j=1}^{p}$. The goal is to characterize an extreme-survival-response support in the predictor space and identify the corresponding box-defined group of samples using a recursive peeling method derived from PRIM.

3.1 Survival Notation and Definitions

The response variable being subject to censoring, we use the general random censoring model. We focus on a univariate right-censored survival outcome under the assumptions of independent observations, non-competitive risks and random (type-I or -II) non-informative censoring. Denote the *true* survival time (or lifetime/failure time) by the random variable T and the *observed* censoring time by the random variable C, then the *observed* survival time is the random variable $Y = \min(T, C)$. Also, under our assumptions, C is assumed to be independent of T conditionally on covariates $\mathbf{X}$. Let the *observed* event indicator random variable be $\varDelta = I(T \leqslant C)$.

Using subscripts $m \in \{1, \ldots, M\}$ and $l \in \{1, \ldots, L\}$ for the covering and box induction/peeling loops, respectively, a peeling at step (m, l) of the box induction/peeling sequence produces a partition of the survival data from the parent box $B_{m,l-1}$ into two partitions for a given set of covariates: the child box $B_{m,l}$ and its complement (for detailed notation, see companion article [5] and Algorithm 1 below). Let's refer to the child box and its complement described above by the "in-box" and the "out-of-box", respectively.

Dropping further step subscripts (m, l) for simplicity, assume that there are n individuals in parent box $B_{m,l-1}$. For each observation $i \in \{1, \ldots, n\}$ in parent box $B_{m,l-1}$, the true survival time, observed censoring time, observed survival time and observed indicator event are the realizations denoted by T_i, C_i, $Y_i = \min(T_i, C_i)$ and $\delta_i = I(T_i \leqslant C_i)$, respectively. Finally, the observed data in parent box $B_{m,l-1}$ consists of $(Y_i, \delta_i, \mathbf{x}_i)$, where $\mathbf{x}_i = [x_{i,1} \ldots x_{i,p}]^T$, for $i \in \{1, \ldots, n\}$.

3.2 Survival-Specific Peeling Rule

We describe in [5] the use of several candidate survival-specific peeling criteria and discuss their merits or strengths. Survival-specific peeling criteria are used to decide which covariate will be selected to give the best peel between two boxes from two consecutive generations (parent-child descendance) of the box induction/peeling loop in a recursive peeling algorithm (see section 3.5).

Briefly, to account for censoring, we simply supervise by proxy for extreme time-to-event outcome, turning the censored outcome y into an uncensored "surrogate" outcome z. The focus is on selecting a sub-box $b_{m,l}$ at step (m, l) of the box induction/peeling sequence that is to be peeled off from the parent box $B_{m,l-1}$ along one of its faces (i.e. direction of peeling := axis of dimension j) to induce the next child box $B_{m,l}$ and its complement. This is done by maximizing the "surrogate" outcome rate of increase between two consecutive generations of boxes $B_{m,l-1}$ and $B_{m,l}$ of the box induction/peeling sequence. Denote by $z(m, l)$ the box "surrogate" outcome at step or generation (m, l) of the box induction/peeling sequence (Algorithm 1). The rate of increase in $z(m, l)$ at step or generation (m, l) between two consecutive generations of boxes $B_{m,l-1}$ and $B_{m,l}$ is defined as:

$$r(m, l) = \frac{z(m, l) - z(m, l-1)}{\beta_{m,l-1} - \beta_{m,l}} \tag{1}$$

Finally, the particular sub-box $b^*_{m,l}$ that is chosen to yield the largest box increase rate $r(m, l)$ between box $B_{m,l-1}$ and the next one $B_{m,l}$ is such that

$$\begin{aligned} B_{m,l} &= B_{m,l-1} \backslash b^*_{m,l} \quad \text{, where} \\ b^*_{m,l} &= \underset{b_{m,l} \in C(b_{m,l})}{\text{argmax}} \; [r(m, l)] \,, \end{aligned} \tag{2}$$

where $C(b_{m,l})$ represents the class of potential sub-boxes $b_{m,l}$ eligible for removal at step (m, l).

3.3 Non-Parametric Survival Peeling Criteria

Currently, our Survival Bump Hunting implementation in our R package `PRIMsrc` [7] offers three statistics as surrogate of outcome $z(m, l)$, at step (m, l) (eq: 1): (i) the Log-Rank Test statistic, denoted $\hat{\chi}_{LRT}(m, l)$, (ii) the Nelson–Aalen Summary statistic, denoted $\hat{\Lambda}_{CHS}(m, l)$ and (iii) the CPH-derived Log Hazard Ratio statistic (assuming proportional hazards), denoted $\hat{\lambda}_{LHR}(m, l)$. Further discussion of the use of the above estimators as well as a few more alternative survival peeling criteria can be found in our companion article, although none of these is preferred nor implemented in our R package (see [5]).

All three peeling criteria statistics can be used to maximize the differences in survival outcomes between two consecutive boxes $\hat{B}_{m,l-1}$ and $\hat{B}_{m,l}$ of the box induction/peeling sequence. This leads to the derivation of the corresponding box rate of increase estimate $\hat{r}(m, l)$, at step (m, l), according to equation 1:

$$\hat{r}_{LRT}(m, l) = \frac{\hat{\chi}_{LRT}(m, l) - \hat{\chi}_{LRT}(m, l-1)}{\hat{\beta}_{m,l-1} - \hat{\beta}_{m,l}} \tag{3}$$

$$\hat{r}_{CHS}(m, l) = \frac{\hat{\Lambda}_{CHS}(m, l) - \hat{\Lambda}_{CHS}(m, l-1)}{\hat{\beta}_{m,l-1} - \hat{\beta}_{m,l}} \tag{4}$$

$$\hat{r}_{LHR}(m, l) = \frac{\hat{\lambda}_{LHR}(m, l) - \hat{\lambda}_{LHR}(m, l-1)}{\hat{\beta}_{m,l-1} - \hat{\beta}_{m,l}} \tag{5}$$

3.4 Box End-Point Statistics

One important application of survival/risk modeling is to identify and segregate samples for predictive diagnostic and/or prognosis purposes. Direct applications include the stratification of patients by diagnostic and/or prognostic groups and/or responsiveness to treatment. Therefore, survival modeling is usually performed to predict/classify patients into risk or responder groups from which one usually derives survival/risk functions estimates. Below is a summary of box end-point statistics of interest one can derive in our Survival Bump Hunting method. Each is defined and cross-validated (see section 4) for each step or generation (m, l). They are all implemented in our R package `PRIMsrc` [7] (see figures in [5]):

1. Log Hazards Ratios (*LHR*), denoted $\lambda(m, l)$ between the highest-risk group/box and lower-risk groups/boxes of the same generation.
2. Log-Rank Test statistic (*LRT*), denoted $\chi(m, l)$ between the highest-risk group/box and lower-risk groups/boxes of the same generation.
3. Concordance Error Rate (*CER*), denoted $\theta(m, l)$ in the highest-risk group/box, that is a prediction performance metric taking censoring into account. For each step (m, l), $\theta(m, l) = 1 - C(m, l)$, where C is Harrel's Concordance Index for censored data [16], a rank correlation U-statistic, to estimate the probability of concordance between predicted and observed survival times.
4. Event-Free Probability (EFP), denoted $P_0(m, l)$ or probability of non-event until a certain time $T(m, l)$ in the highest-risk group/box. If $P_0(m, l)$ is not reached for a specified time $T(m, l)$, we determine the limit end-point $P'_0(m, l)$ or Minimal Event-Free Probability (*MEFP*) and corresponding maximal time $T'(m, l)$, which are always observable.
5. Event-Free Time (EFT), denoted $T_0(m, l)$ or time to reach a certain end-point probability $P(m, l)$ in the highest-risk group/box. If $T_0(m, l)$ is not reached for a certain probability $P(m, l)$, we determine the limit end-point $T'_0(m, l)$ or Maximal Event-Free Time (*MEFT*) and corresponding minimal probability $P'(m, l)$, which are always observable.
6. Box characteristics: $2p$ box edges $\left[t_j^-(m, l), t_j^+(m, l)\right]_{j=1}^{p}$, box support (mass) $\beta(m, l)$ and box membership indicator $\boldsymbol{\gamma}(m, l)$.
7. Traces of Covariate Usage $VU(m, l)$ and Covariate Importance $VI(m, l)$
8. Kaplan–Meir curves of survival probability values with log-rank test permutation p-values $p(m, l)$

3.5 Estimation by Patient Recursive Survival Peeling

The strategy employed here is a recursive peeling algorithm for survival bump hunting. Our "Patient Recursive Survival Peeling" method proceeds similarly as to which is done in PRIM except for the box induction peeling/pasting criteria and the induction stopping rule (see section 2.4):

Algorithm 1 Patient Recursive Survival Peeling (annotated below w.l.o.g for a maximization problem). See detailed notation in companion article [5].

```
• Start with the training data $\mathcal{L}_{(1)}$ and a maximal box $\hat{B}_1$ containing it
• For $m \in \{1, \ldots, M\}$:
    1: Generate a box $\hat{B}_m$ using the remaining training data $\mathcal{L}_{(m)}$
    2: For $l \in \{1, \ldots, L\}$:
        – Top-down peeling: Generate a box $\hat{B}_{m,l}$ by conducting a stepwise covariate selection/usage: shrink the box by compressing one face (peeling), so as to peel off a quantile $\alpha_0$ of observations of a covariate $\mathbf{x}_j$ for $j \in \{1, \ldots, p\}$. Choose the direction of peeling $j$ that yields the largest box increase rate $\hat{r}(m,l)$ of the statistic used as peeling criterion between box $\hat{B}_{m,l-1}$ and $B_{m,l}$ in the next generation: Log-Rank Test $\hat{\chi}_{LRT}(m,l)$, Cumulative Hazard Summary $\hat{\Lambda}_{CHS}(m,l)$, Log Hazards Ratio $\hat{\lambda}_{LHR}(m,l)$. The current box $\hat{B}_{m,l-1}$ is then updated: $\hat{B}_{m,l} = \hat{B}_{m,l-1} \backslash \hat{b}^*_{m,l}$, where $\hat{b}^*_{m,l} = \underset{\hat{b}_{m,l} \in C(b_{m,l})}{\text{argmax}} \; [\hat{r}(m,l)]$
        – Bottom-up pasting: Expand the box along any face (pasting) as long as the resulting box increase rate $\hat{r}(m,l) > 0$
        – Stop the peeling loop until a minimal box support $\hat{\beta}_{m,L}$ of $\hat{B}_{m,L}$ is such that it reached a minimal box support $0 \leqslant \beta_0 \leqslant 1$, expressed as a fraction of the data: $\hat{\beta}_{m,L} \leqslant \beta_0$
        – $l \leftarrow l + 1$
    3: Step #2 give a sequence of nested boxes $\{\hat{B}_{m,l}\}_{l=1}^{L}$, where $L$ is the estimated number of peeling/pasting steps with different numbers of observations in each box. Call the next box $\hat{B}_{m+1} = \hat{B}_{m,L}$. Remove the data in box $\hat{B}_m$ from the training data: $\mathcal{L}_{(m+1)} = \mathcal{L}_{(m)} \backslash \hat{B}_m$
    4: Stop the covering loop when running out of data or when a minimal number of observations remains within the last box $\hat{B}_M$, say $\hat{\beta}_M \leqslant \beta_0$
    5: $m \leftarrow m + 1$
• Steps #1 – #5 produce a sequence of (not necessarily nested) boxes $\{\hat{B}_m\}_{m=1}^{M}$, where $M$ is the estimated total number of boxes covering $\mathcal{L}_{(1)}$
• Collect the decision rules of all boxes $\{\hat{B}_m\}_{m=1}^{M}$ into a simple final decision rule $\hat{\mathcal{R}}$ of the solution region $\hat{R}$ of the form: $\hat{\mathcal{R}} = \bigcup_{m=1}^{M} \hat{\mathcal{R}}_m$, where $\hat{\mathcal{R}}_m = \bigcap_{j \in J}(\mathbf{x}_j \in [t^-_{j,m}, t^+_{j,m}])$ giving a full description of the estimated bumps in the entire input space
```

4. Cross-Validation for Recursive Peeling Methods and a Survival/Risk Outcome

4.1 Setup

Cross-validation of box estimates should include all steps of the box generation sequence $\{B_m\}_{m=1}^{M}$ i.e. for the (outer) coverage loop of our "Patient Recursive Survival Peeling" method (Algorithm 1), each step of which involves a peeling sequence $\{B_{m,l}\}_{l=1}^{L}$ of the (inner) box peeling/induction loop. However, for simplicity, cross-validation designs of box estimates and resulting decision rule $\hat{\mathcal{R}}_m$ are described for *fixed* $m \in \{1, \ldots, M\}$ of the complete box sequence $\{\{\hat{B}_{m,l}\}_{l=1}^{L}\}_{m=1}^{M}$.

Although using a fully independent test set for evaluating a predictive bump hunting model is always advisable, the sample size in discovery-based studies is often too small to effectively split the data into training and testing sets (Split-Sample-Validation) and provide accurate estimates [2, 11, 24]. In such cases, resampling techniques such as K-fold Cross-Validation (CV) are required [1, 22].

4.2 K-fold Cross-Validation

4.2.1 Resampling Design - Notation

In resampling based on full K-fold cross-validation, the whole data $\mathcal{L}$ is randomly partitioned into K approximately equal parts of test samples or test-sets $(\mathcal{L}_1, \ldots, \mathcal{L}_k, \ldots, \mathcal{L}_K)$. For each test-set $\mathcal{L}_k$, for $k \in \{1, \ldots, K\}$, a training set $\mathcal{L}_{(k)}$ is formed from the union of the remaining $K - 1$ subsets: $\mathcal{L}_{(k)} = \mathcal{L} \setminus \mathcal{L}_k$. The process is repeated K times, so that K test-sets $\mathcal{L}_k$ are formed of about equal size and K corresponding training subsets $\mathcal{L}_{(k)}$, for $k \in \{1, \ldots, K\}$. Typically, $K \in \{3, \ldots, 10\}$. The training samples are approximately of size $\approx n(K - 1)/K$ and the test samples are of size $n^t \approx n/K$.

4.2.2 Cross-Validation Techniques

Although resampling methods are useful in assessing the prediction accuracy of classifiers, they are not directly applicable for predictive survival modeling applications [12, 22, 26, 27]. There are remaining issues to deal with K-fold CV: first, how to cross-validate a simple peeling trajectory $\{\hat{B}_l\}_{l=1}^{L}$ and related statistics is not straightforward; second, how to cross-validate survival curve estimates and related statistics is also not intuitive (see also [25]). So, regular K-fold cross-validation is not directly applicable to the joint task of box decision rules making by recursive peeling and survival estimation. One must design a specific cross-validation technique(s) of survival bump hunting that is amenable to this joint task. Recently, we described a cross-validation technique for recursive peeling methods in a survival/risk setting [4, 5].

Briefly, we proposed two techniques by which K-fold cross-validation estimates can be computed (see details in companion article [5]):

- *Averaging Technique*: Estimations are first computed for each “in-box” test subset samples, then averaged over the cross-validation loops of random splitting to give the “Averaged Cross-Validation” estimates.
- *Combining Technique*: All “in-box” test subset samples are first collected from all the cross-validation loops of random splitting to build a *combined* test “in-box” and corresponding *combined* test “in-box” samples to compute *once* the final “Combined Cross-Validation” estimates.

To account for the high variability of cross-validated estimates [11, 13, 21], we iterate each cross-validation procedure several times over some replicates B (typically, $B \geqslant 10$) to average the estimates and reduce their variance and generate “Replicated Cross-Validated” estimates. Note that, unlike in the averaging technique, cross-validated combined estimates are computed on test samples of size n instead of $n^t \approx n/K$, which could be an advantage in the case of tiny sample size n.

4.2.3 Model Peeling Length Optimization Criterion

In model tuning/selection, there is a trade-off between under-fitting and over-fitting that is achieved by optimizing an empirical function, taking censoring into account, or objective criterion that we call “optimization criterion”. The one that we derive is adapted to the task of cross-validation of a survival bump hunting model fit by a recursive peeling method with a survival outcome [5]. Specifically, we train a peeling model by optimizing its complexity, that is here, the length or number of peeling steps of the peeling sequence (or peeling trajectory profiles).

Cross-validated estimates of box end-points statistics (described in 3.4) are computed using the left-out test-set $\mathcal{L}_k$. Three of these are the Log Hazard Ratio (*LHR*), Log-Rank Test (*LRT*) and the cross-validated estimate of prediction performance, namely the Concordance Error Rate (*CER*) that is obtained by calculating the test-set error rate using the left-out test-set $\mathcal{L}_k$. For details on how the three optimization criteria are derived and used, see companion article [5].

5. Features

5.1 Description and Current Scope

`PRIMsrc` comes as a user-friendly open source software with a complete documentation. The current stable version is available as an R package (0.6.0) on the CRAN repository at `https://cran.r-project.org/web/packages/PRIMsrc/index.html`. `PRIMsrc` allows open source development via GitHub at `https://github.com/jedazard/PRIMsrc` and testing via TravisCI at `https://travis-ci.org/`. The current version of the software is a development release that only implements the case of a survival response. Survival Bump Hunting (SBH) in `PRIMsrc` is also restricted to a *directed* peeling search of the first box covered by the recursive coverage (outer) loop of our Patient Recursive Survival Peeling (PRSP) algorithm (see Algorithm 1). Finally, the main function relies on an optional variable pre-selection procedure that is done in this version by a cross-validated penalization of the partial likelihood using the R package `glmnet`.

5.2 Development Branches:

Implementations of complete Survival Regression Classification or SRC features are being implemented and will be added soon. The development release is available on GitHub. It is organized in three branches under development:

1. Branch #0 (master - default) containing the most up-to-date version 0.6.1 to eventually merge with branch #1 then #2;
2. Branch #1 (devel) containing version 0.7.0. The specific treatment of dimension-reduction or variable selection in high-dimensional settings is implemented differently in the previous GitHub branch (0.6.1) or CRAN version (0.6.0) than in this branch (0.7.0). In the former, the main function relies on an optional variable pre-selection procedure that is run before the PRSP algorithm. At this point, this is done by a cross-validated penalization of the partial likelihood using the R package `glmnet`. In the latter, we implemented an integrated cross-validation procedure that involves a univariate Survival Bump Hunting (SBH) variable selection strategy. Ultimately, this will provide a more rigorous treatment of model validation, a better control on the user-end and an improvement of the maintenance on the back-end.
3. Branch #2 (unified) containing version 1.0.0. The latter branch will host the future complete version of the code, including undirected peeling search by Patient Rule Induction Method (PRIM) that will allow the unified treatment of bump hunting for every type of continuous, discrete and censored responses.

5.3 Package Main Features

The following describes the end-user functions that are needed to run a complete procedure. The other internal subroutines are not documented in the manual and are not to be called by the end-user at any time. For computational efficiency, `PRIMsrc` offers a parallelization option that is done by passing a few parameters needed to configure a cluster. This is indicated by an asterisk (*). At this point, the end-user R features are categorized as follows (see documentation in our R package for details [7]):

1. SURVIVAL BUMP HUNTING FUNCTION
   `sbh()` (*) : Main end-user function for fitting a cross-validated Survival Bump Hunting (SBH) model and returning cross-validated estimates of end-points statistics of interest (see 3.4). It has various options to allow cross-validation procedures to control model size (# covariates) and model complexity (# peeling steps). There is also an option for efficient statistical computing by parallel computation, is also available to allow: (i) user-defined choice of type of process communication mechanisms (MPI, SOCKET and PVM), (ii) parallel random number generation and (iii) seed

management for reproducible results / research

2. S3-GENERIC FUNCTIONS FOR SUMMARY, DISPLAY, PLOT AND PREDICTION
`summary()` : S3-generic summary function to summarize the main parameters used to generate the `PRSP` object.
`print()` : S3-generic print function to display the cross-validated estimated values of the `PRSP` object.
`plot()` : S3-generic plotting function for two-dimensional visualization of original or predicted data scatter as well as cross-validated box vertices of a `PRSP` object.
`predict()`: S3-generic predict function to predict the box membership and box vertices on an independent set.

3. PLOTTING FUNCTIONS FOR MODEL VALIDATION AND VISUALIZATION OF RESULTS
`plot_profile()` : Function for plotting the cross-validated tuning profiles of a `PRSP` object.
`plot_boxtraj()` : Function for plotting the cross-validated peeling trajectories/profiles of a `PRSP` object.
`plot_boxtrace()` : Function for plotting the cross-validated covariates traces of covariate importance and covariate usage of a `PRSP` object.
`plot_boxkm()` : Function for plotting the cross-validated survival distributions of a `PRSP` object.

4. DATASETS
`Synthetic.1, Synthetic.1b, Synthetic.2, Synthetic.3, Synthetic.4:` Five datasets from simulated regression survival models #1-4 as described in companion article [5], representing low- and high-dimensional situations, and where regression parameters represent various types of relationship between survival times and covariates including saturated and noisy situations.
`Real.1, Real.2` : Two publicly available datasets: one HIV clinical data from the Women's Interagency HIV cohort Study (WIHS), one lung cancer genomic data from the Chemores Cohort Study, representing low-dimensional ($p < n$), or high-dimensional ($p \gg n$) cases, respectively.

5. FUNCTION FOR PACKAGE NEWS
`PRIMsrc.news()`

5.4 Simulation Design

The p-dimensional covariates $\mathbf{x}_i = [x_{i,1} \ldots x_{i,p}]^T$, for $i \in \{1, \ldots, n\}$, were drawn identically and independently (i.i.d) from a p-multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and variance-covariance matrix $\boldsymbol{\Sigma}$: $\mathbf{x}_i \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Simulated realizations of true survival times T_i were were i.i.d. sampled from an exponential distribution with rate parameter λ (and mean $\frac{1}{\lambda}$): $T_i \sim \text{Exp}(\lambda)$. Simulated realizations C_i of true censoring times were i.i.d. sampled from a uniform distribution: $C_i \sim U(0, v)$ with $v > 0$, so that approximately $100 \times \pi(\%)$ of the simulated realizations of observed survival times $Y_i = \min(T_i, C_i)$ were censored, where $\pi \in \{0.3, 0.5, 0.7\}$. Finally, the simulated realizations of observed event (non-censoring) random variable indicator were as follows: $\delta_i = I(T_i \leqslant C_i)$.

In summary, our simulation was done using the following parameters: $n = 250$ and $p = 2$ (bivariate normal distribution), without inter-covariate correlation: $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$, with $\boldsymbol{\mu} = [5,\ 5]^T$; by characterization of the first coverage box B_1 (i.e. for $m = 1$), using constrained/directed peeling, without pasting and with default meta-parameter values $(\alpha_0, \beta_0) = (0.10, 0.05)$; with censoring rate π = 0.5, in a single regression survival model with regression parameter $\boldsymbol{\eta} = [2,\ -4]^T$, representing a low-dimensional saturated model; using $K = 5$-fold cross-validation, $A = 1024$ for the permutation p-values and $B = 128$ independent replications.

5.5 Summary of Plot Outputs

We explain and illustrate below how to use the S3-generic plotting function (`plot`) and the four specific plotting functions (`plot_profile`, `plot_boxtraj`, `plot_boxtrace`, `plot_boxkm`) from our R package `PRIMsrc` (see also [5, 7] for more details).

5.5.1 Data Scatter Plots

`plot` is an S3-generic plotting function that provides a two-dimensional visualization of data scatter of the original or predicted data as well as cross-validated box vertices of a `PRSP` object. The scatter plot is for a given peeling step of the peeling sequence and a given plane, both specified by the user. A peeling step includes step #0 corresponding to the situation where the starting box covers the entire test-set data $\mathcal{L}_k$ before peeling (Algorithm 1). Figure 1 illustrates this for the original dataset ($n = 250$) described in our simulation design (above section 5.4) and predictions made on a new dataset of $n = 100$ observations assumed from the same population, that is, drawn i.i.d. from the same distribution.

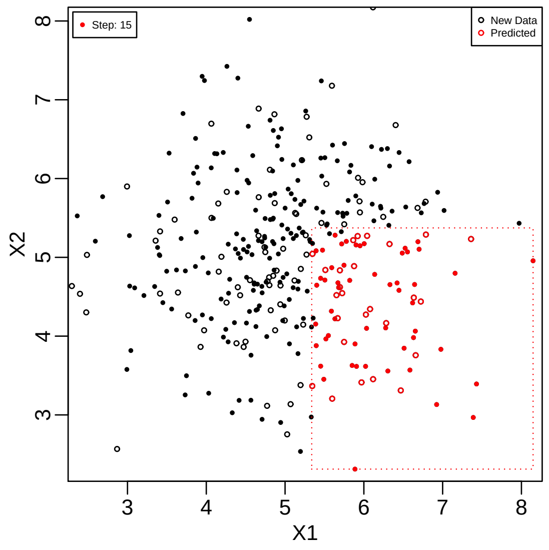

Figure 1: *Scatter plot of original and predicted points from simulated data (section 5.4). The original data points are shown ($n = 250$, filled dots) along with the new data points ($n = 100$, empty dots). Note how all predicted data points (empty red dots) fall within the boundaries of the "in-box" found by the PRSP algorithm. Results are for the optimal peeling length $\bar{L}^{rcv} = 14$ of the cross-validated peeling trajectory (i.e. #peeling steps = 15, counting step #0), for the "Replicated Combined CV" (RCCV) technique and the Cumulative Hazard Summary (CHS) and Log-Rank Test (LRT) statistics used as peeling and optimization criterion, respectively.*

5.5.2 Cross-Validated Tuning Profiles

`plot_profile` provides cross-validated tuning profiles of the box end-points statistics (section 3.4) Log Hazard Ratio (*LHR*), Log-Rank Test (*LRT*) or Concordance Error Rate (*CER*), depending on the optimization criterion chosen, as a function of peeling length or peeling steps of the peeling trajectory (model complexity). A peeling step includes step #0 corresponding to the situation where the starting box covers the entire test-set data $\mathcal{L}_k$ before peeling (Algorithm 1). These statistics are internally or interactively used to get the "Replicated CV" optimal length of the peeling trajectory: $\bar{L}^{rcv}$ (Figure 2). In order to successfully determine the profiles minimizer or maximizer (section 4.2.3), the cross-validated tuning profile should be approximately non-monotone

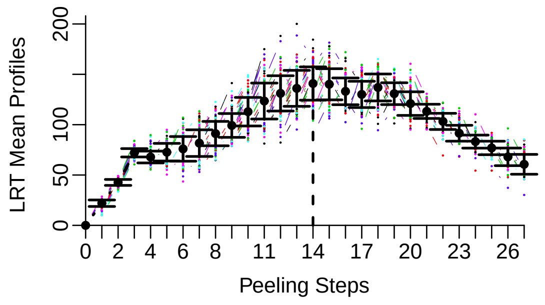

Figure 2: *Typical successful cross-validated tuning profile for the simulated data. Peeling stops at a "Replicated CV" optimal peeling length $\bar{L}^{rcv} = 14$, (i.e. #peeling steps = 15, counting step #0) that is reached within the boundaries of possible peeling lengths (i.e. between 1 and the maximal peeling length $\bar{L}^{rcv}_m 1 = 27$). Results are for the "Replicated Combined CV" (RCCV) technique and for the Cumulative Hazard Summary (CHS) and the Log-Rank Test (LRT) statistics used as peeling and optimization criterion, respectively. $\bar{L}^{rcv}$ is shown with the vertical black dashed line. Each colored profile corresponds to one of the replications ($B = 128$). The cross-validated mean profile of the statistic used in the optimization criterion is shown by the dotted black line with standard error of the sample mean.*

up to sampling variability (see figures in [5]). In addition, one expects an inflation of variance of cross-validated point estimates towards the right-end of the cross-validated tuning profile corresponding to an increase in overfitting and model uncertainty for more complex models (Figure 2).

5.5.3 Peeling Trajectories

`plot_boxtraj` provides cross-validated box peeling trajectories, estimated by step functions of the covariates box cuts as a function of box support/mass (Figures 3). They are read from right to left as they track the top-down peeling of the box induction process (peeling loop) of our "Patient Recursive Survival Peeling" method (Algorithm 1). These trajectories are, up to sampling variability:

- Monotone (increasing or decreasing) functions for each input covariate $\mathbf{x}_j$, for $j \in \{1, \ldots, p\}$.
- Non-monotone (increasing then decreasing) functions for *LHR* $\bar{\lambda}^{rcv}(l)$.
- Non-monotone (increasing then decreasing) functions for *LRT* $\bar{\chi}^{rcv}(l)$.
- Non-monotone (decreasing then increasing) functions of *CER* $\bar{\theta}^{rcv}(l)$.
- Monotone decreasing functions for *MEFP* $\overline{P_0'}^{rcv}(l)$.
- Monotone decreasing functions for *MEFT* $\overline{T_0'}^{rcv}(l)$.

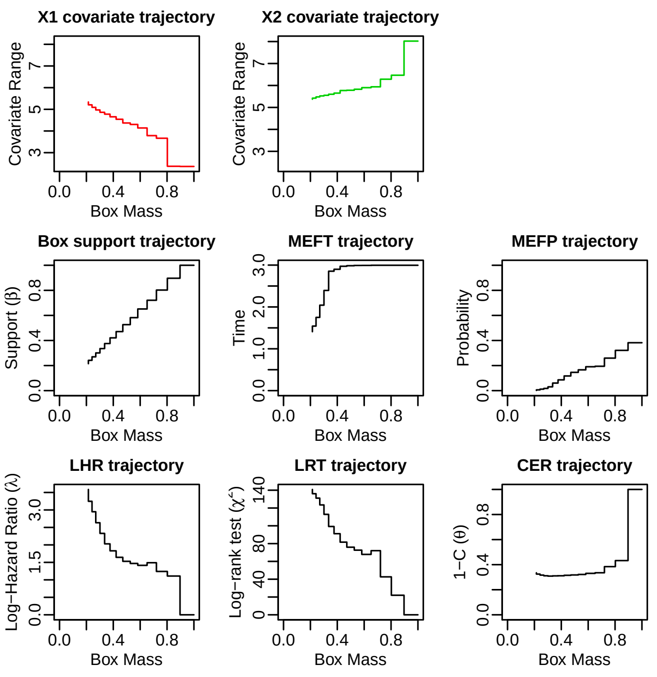

Figure 3: *Cross-validated peeling trajectories for the simulated data described in section 5.4. Results are for the "Replicated Combined CV" (RCCV) technique and for the Cumulative Hazard Summary (CHS) and the Log-Rank Test (LRT) statistics used as peeling and optimization criterion, respectively.*

5.5.4 Trace Curves

`plot_boxtrace` provides cross-validated trace curves of covariate importance and covariate usage, estimated by piece-wise linear and step functions, respectively, as a function of box support/mass (Figures 4). Similarly to peeling trajectories, they are read from right to left. Trace curves of covariate importance show on a single plot: (i) the amplitude of used covariates, (ii) the order (prioritization) with which these covariates are used, and (iii) the extent of the number of peeling steps by which each covariate is used. Covariate traces are reminiscent of the concept of variable selection from the fields of decision tree and regularization (see [5] for more details and references).

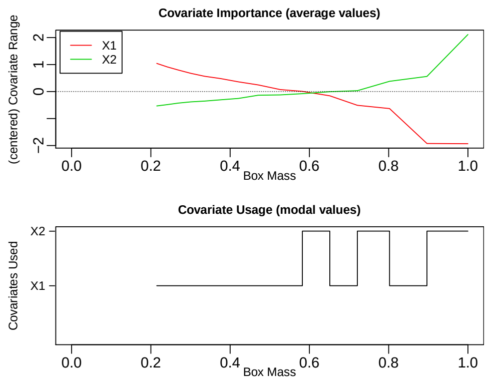

Figure 4: *Cross-validated trace plots of Covariate Importance $\overline{VI}(l)$ (top) and Covariate Usage $\overline{VU}(l)$ (bottom) for the simulated data described in section 5.4. Results are for the "Replicated Combined CV" (RCCV) technique and for the Cumulative Hazard Summary (CHS) and the Log-Rank Test (LRT) statistics used as peeling and optimization criterion, respectively.*

5.5.5 Survival Curves

`plot_boxkm` provides a series of subplots corresponding to the cross-validated number of peeling steps of our Patient Recursive Survival Peeling method. Each subplot shows the cross-validated Kaplan–Meir estimates of the survival functions, as a function of survival time, of both “in-box” (red) and “out-of-box” (black) samples, corresponding respectively to the high-risk and low-risk groups (Figure 5). Each subplot also displays the corresponding step number along with cross-validated Log Hazard Ratio (*LHR*), Log-Rank Test (*LRT*) and log-rank permutation p-value $\tilde{p}^{cv}(l)$ of survival distribution separation (see details in companion article [5]). A single survival curve always exists at Step #0 corresponding to the situation where the starting box covers the entire test-set data $\mathcal{L}_k$ before peeling (5).

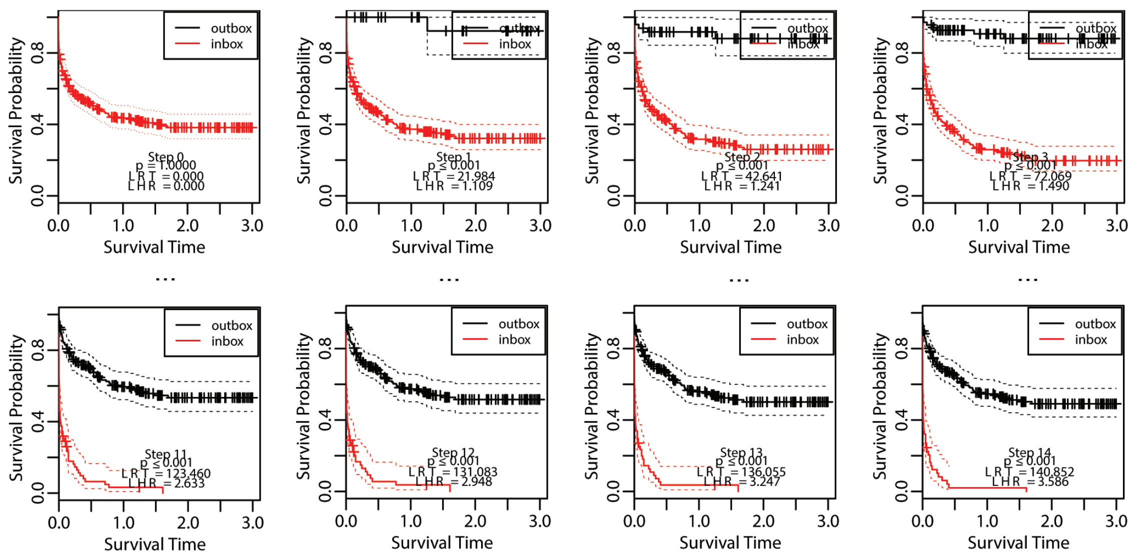

Figure 5: *Cross-validated Kaplan–Meir survival probability curves of the high-risk (red curve “in-box”) and low-risk (black curve “out-of-box”) groups for the simulated data described in section 5.4. Results are for the “Replicated Combined CV” (RCCV) technique and for the Cumulative Hazard Summary (CHS) and the Log-Rank Test (LRT) statistics used as peeling and optimization criterion, respectively. For conciseness, only the first four and last four steps are shown. Cross-validated LRT, LHR and permutation p-values $\tilde{p}^{cv}(l)$ of “in-box” samples are shown at the bottom of the plot with the corresponding peeling step. P-values are bounded by the precision limit. Notice that the single survival curve at Step #0 corresponding to the situation before peeling (5) and how the survival curves of “in-box” and “out-of-box” samples separates as the peeling progresses.*

6. Implementation Design

6.1 Computational Complexity Considerations

The “Patient Recursive Survival Peeling” (PRSP) algorithm, being based on PRIM (see section 3.5) is by design a less “greedy” recursive (peeling) algorithm as compared to a similar recursive (partitioning) algorithm as CART. The reason is that, in CART, the data splits at an exponential rate as the space undergo partitioning (typically by binary splits) as opposed to a more patient rate in decision boxes (typically by controlled quantile). In this sense, bump hunting by recursive peeling may be a more efficient way to learn from the data.

Specifically, using previous notations, CART takes $L_{CART}(n) = \log_2(n) - 1$ split steps with splits of equal size, whereas PRIM takes at most $L_{PRIM}(\alpha_0, \beta_0) = \left\lceil \frac{\log(\beta_0)}{\log(1-\alpha_0)} \right\rceil$ peeling steps for peeling meta-parameters α_0 and β_0 (see section 2.4 and [15] for details). In fact, the maximal number of peeling steps is achieved when one observation is peeled at each step, i.e. when $\alpha_0 = \frac{1}{n}$ and when the box support reduces to a single data point, i.e. when $\beta_0 = \frac{1}{n}$; in which case, $L_{PRIM}(n) = \left\lceil \frac{\log(\frac{1}{n})}{\log(1-\frac{1}{n})} \right\rceil$. For example, with $n = 128$, $L_{CART}(n) = 6 \ll L_{PRIM}(n) = 619$.

Considering the trained box $\hat{B}_{m,l(k)}$ at step (m, l) of the box covering/induction sequence that is constructed from training set $\mathcal{L}_{(k)}$ of sample size, denoted here n (see sections 4.2.1), where M, L are the number of covering and peeling/pasting steps of the backfitting algorithm, respectively (see section 3.5), it can be shown that the computational complexity of finding the complete box sequence $\{\{\hat{B}_{m,l}\}_{l=1}^{L}\}_{m=1}^{M}$ by PRIM is of the order $O(MLpn\log(n))$ [8, 15, 17]. So, the overall complexity of a B-replicated Survival Bump Hunting by our PRSP algorithm is of the order $O(BMLpn\log(n))$, which can be prohibitive for large B.

6.2 Code Profiling

Code profiling can be used to identify bottlenecks in R code that could benefit e.g. from being replaced by compiled code. The command `utils::Rprof()` is used to control R code profiling (including compiled code) on Windows, OSX and most Unix-alike platforms. Below, serial computation times of the main function `sbh()` are shown for a mid-size ($n = 200, p = 100$) synthetic dataset (similarly to the simulation design 5.4) without replications and computation of permutation p-values (Table 1).

Table 1: *Code profiling results for a serial computation (on a single CPU core) of the main function `sbh()` on a single ($n = 200, p = 100$) simulated dataset, for $B = 1$, without computation of permutation p-values. Results are broken down by total CPU times of internal subroutines, including `R` and `Fortran` codes, ordered by decreasing CPU times and reduced to the top 12 for conciseness. Tested under version 0.6.0 of `PRIMsrc`.*

PRIMsrc Subroutine	Internal Dependencies	total time (s)	total time (%)
`sbh()`		15.639	99.54
`cv.presel()`		10.933	69.58
	`.Fortran`	10.877	69.23
	`cv.glmnet`	10.865	69.15
	`glmnet`	10.855	69.09
	`coxnet`	10.851	69.06
`cv.comb.box()`		4.571	29.09
`cv.box.rep()`		4.571	29.09
`cv.comb.fold()`		4.239	26.98
`cv.comb.peel()`		4.239	26.98
`peel.box()`		4.234	26.95
	`survdiff()`	3.457	22.00

Clearly, code profiling reveals that, when dealing with the dimensionality of the data, the most expensive computation step ($> 69\%$) comes from our R function `cv.presel()` for covariates pre-selection, done, at this point, by calling the internal dependency subroutine `cv.glmnet()` of the `glmnet` R package (see section 5.3 and documentation of `PRIMsrc` for details [7]).

6.3 Computational Parallelization

R is single threaded, but if one has access to a cluster and/or a computer with multiple cores, one can parallelize the code execution by running several replicated R session simultaneously on more than one core, simply by distributing the tasks to multiple cores on a local or remote machine. Our unique end-user function `sbh()` uses the R package `parallel`, which is designed to build real and virtual clusters and allow users to create a parallel backend, enabling parallel execution and scaling up with the number of available cores (denoted C).

R package `parallel` also allows our R package `PRIMsrc` to include a user-defined choice of type of job communication mechanisms (MPI, SOCKET and PVM) and to implement parallel Random Number Generation (RNG) with seed management, which requires the creation of separate streams of parallel RNG per core, that is done internally by distributing the stream states to the cores (see documentation in [7] for details and examples).

Parallel computing is ideal for embarrassingly parallel tasks such as resampling loops. Computational considerations indicate that the number of replications B can easily be parallelized by distributing one (or more) replication by core. In case replications *and* permutation p-values are desired (e.g. `A`=1024 and `pval=TRUE` in the `sbh()` function), the use of the cluster is definitely recommended. Note that the parallelization is not nested at this point, that is, only the replications are, not the permutations.

6.4 Computational Scalability

The easiest way to quantify the performance of an R function on Windows, OSX and most Unix-alike platforms is to use the command `base::system.time()`. Here, we compared computation times of parallel and serial calculations of the main function `sbh()` on various sizes ($n \in \{200, 300, 400, 500\}$, $p \in \{10, 10^2, 10^3, 10^4\}$) of synthetic datasets (similarly to the simulation design 5.4) for varying numbers of replications ($B \in \{8, 16, 32, 64, 128\}$) and numbers of spawned cores ($C \in \{1, 8, 16, 32, 64, 128\}$), without computation of permutation p-values.

From the computation times output, we get the amount of scalability while keeping balanced loads. Generally, using configuration with larger number of CPU cores does not necessarily result in better parallel computing performance due to threshold effects and computation/memory overhead issues.

The threshold effect is encountered as an effect of distributing processes (parallelization of a number of tasks) relatively to the number of cores. See the observed scaling threshold due to the requested number B of replications relative to the number C of spawned cores in Figure 6 (left). Here, with the parallelization done on the number of replications B and for a given number of sample n and covariates p, the total elapsed time is divided proportionally to the number of spawned CPU cores (C). However, once the number of spawned cores C exceeds the number of replications (one per core), adding more cores will not reduce the computation time any further. For instance, given a large number of spawned cores $C = 128$, suppose the number of replications $B = 64$, that is, less than the number of spawned cores C, one sees that the computation time decreases from $C = 1$ to $C = 64$, but not any further from $C = 64$ to $C = 128$.

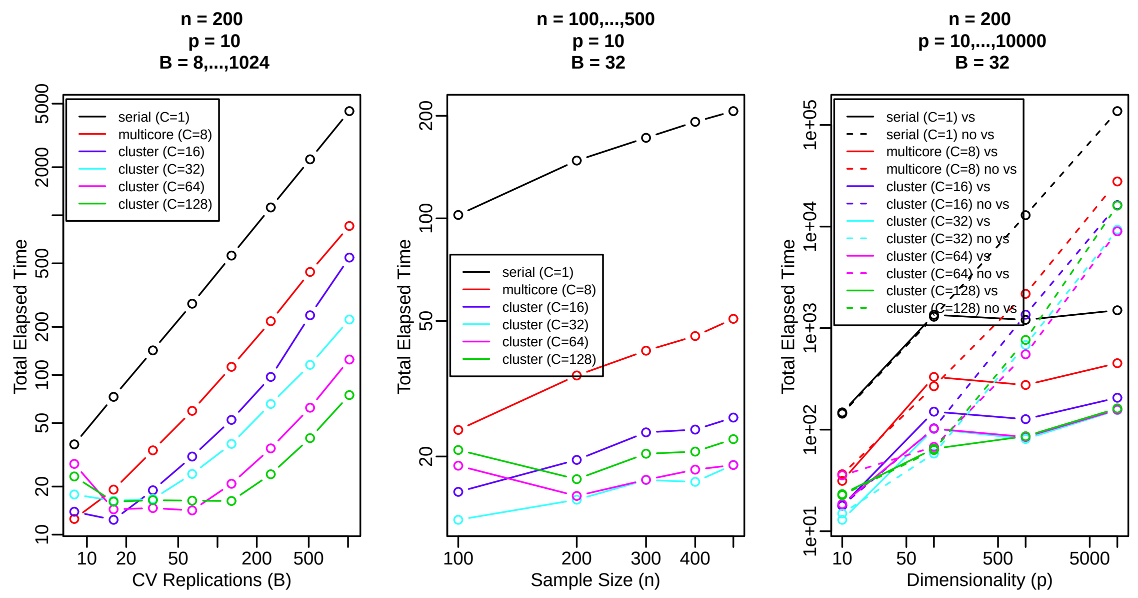

Figure 6: *The total computation time is plotted as function of: (i) the number of replications B, for fixed sample size $n = 200$ and dimensionality $p = 10$ (left), or (ii) the sample size n for fixed replications $B = 32$ and dimensionality $p = 10$ (center), or the dimensionality p for fixed sample size $n = 200$ and replications $B = 32$ (right), for various configuration of cluster size (number C of spawned cores). "vs" stands for variable selection. All axes are on a log scale. Tested under version 0.6.0 of* `PRIMsrc`.

The overhead issues are generally encountered when the individual task does not take a significant amount of time and memory (RAM) in the first place. With individual tasks requiring little computation and memory, the overhead of: (i) scheduling and distributing the tasks to processing units, (ii) communicating between processes, and (iii) gathering and returning the results, can be greater than the time to execute the tasks in a serial manner. This is illustrated in the scalability performance plots of Figure 6:

given a fixed number of replications (e.g. $B = 32$), scaling up from $C = 32$ to $C = 64$ or even $C = 128$ degrades the gains in performance if the individual task size is relatively small, e.g. $n = 100$ and $p = 10$ (Figure 6, center), or e.g. $n = 200$ and $p = 10 - 100$ (Figure 6, right).

Dimension reduction by variable selection significantly improves the computation time: compare for instance the solid versus dashed lines of the same color in Figure 6 (right). But the implemented variable selection procedure seems to plateau at $p \approx 100$ selected variables for unknown reasons.

7. Acknowledgments

J-E. Dazard and J. Sunil Rao were supported in part by National Institute of Health (NIH) grant NCI R01-CA160593A1. Additional support came from NIH grant NCI P30-CA043703 of the Comprehensive Cancer Center at Case Western Reserve University (J-E. D). This work made use of the High Performance Computing Cluster for Advanced Research Computing at Case Western Reserve University.

REFERENCES

[1] Ambroise, C. and McLachlan, G. J. (2002), "Selection bias in gene extraction on the basis of microarray gene-expression data," *Proc Natl Acad Sci U S A*, 99, 6562–6.

[2] Baker, S., Kramer, B., and Srivastava, S. (2002), "Markers for early detection of cancer: Statistical guidelines for nested case-control studies," *BMC Medical Research Methodology*, 2, 4.

[3] Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984), *Classification and Regression Trees*, The Wadsworth statistics/probability series, Boca Raton, FLorida: Chapman and Hall/CRC.

[4] Dazard, J.-E., Choe, M., LeBlanc, M., and Rao, J. (2014), "Cross-Validated Survival Bump Hunting using Recursive Peeling Methods," in *JSM Proceedings. Section for survival methods for risk estimation/prediction*, Boston, MA, USA.: American Statistical Association, vol. IMS JSM, pp. 3366–3380.

[5] — (2015), "Cross-validation and Peeling Strategies for Survival Bump Hunting using Recursive Peeling Methods," *Statistical Analysis and Data Mining*, –.

[6] — (2016), "Survival Bump Hunting For Identification and Characterization of Informative Prognostic Subgroups," (in prep), https://github.com/jedazard/PRIMsrc/blob/master/inst/doc/Abstract.JCB2016.pdf.

[7] Dazard, J.-E., Choe, M., LeBlanc, M., and Santana, A. (2015), "Contributed R Package PRIMsrc for Bump Hunting by Patient Rule Induction Method in Survival, Regression and Classification settings," The Comprehensive R Archive Network, https://cran.r-project.org/web/packages/PRIMsrc/index.html.

[8] Dazard, J.-E. and Rao, J. (2010), "Local Sparse Bump Hunting," *J. Comp. Graph. Statist.*, 19, 900–929.

[9] Diaz-Pachon, D., Dazard, J.-E., and Rao, J. (2015), "Unsupervised bump hunting using principal components," *(submitted)*, –.

[10] Diaz-Pachon, D., Rao, J., and Dazard, J.-E. (2015), "On the explanatory power of principal components," *(submitted)*, –.

[11] Dobbin, K. and Simon, R. (2007), "Sample size planning for developing classifiers using high dimensional DNA microarray data," *Biostatistics*, 8, 101–117.

[12] Dupuy, A. and Simon, R. (2007), "Critical Review of Published Microarray Studies for Cancer Outcome and Guidelines on Statistical Analysis and Reporting," *J. Nat. Cancer Institute*, 99, 147–157.

[13] Efron, B. (1983), "Estimating the error rate of a predication rule: Improvement on cross-validation," *J Amer Stat Assoc*, 78, 316–331.

[14] Fan, J. and Lv, J. (2008), "Sure independence screening for ultrahigh dimensional feature space," *J R Statist Soc B*, 70, 849–911.

[15] Friedman, J. and Fisher, N. (1999), "Bump hunting in high-dimensional data," *Statistics and Computing*, 9, 123–143.

[16] Harrell, F. E., J., Califf, R. M., Pryor, D. B., Lee, K. L., and Rosati, R. A. (1982), "Evaluating the yield of medical tests," *JAMA : the journal of the American Medical Association*, 247, 2543–6.

[17] Hastie, T., Tibshirani, R., and Friedman, J. (2009), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, New York: Springer Science.

[18] Il-Gyo, C. and Chi-Hyuck, J. (2008), "Flexible patient rule induction method for optimizing process variables in discrete type," *Expert Syst. Appl.*, 34, 3014–3020.

[19] LeBlanc, M., Jacobson, J., and Crowley, J. (2002), "Partitioning and peeling for constructing prognostic groups," *Stat Methods Med Res*, 11, 247–74.

[20] LeBlanc, M., Moon, J., and Crowley, J. (2005), "Adaptive Risk Group Refinement," *Biometrics*, 61, 370–378.

[21] Markatou, M., H., T., S., B., and G., H. (2005), "Analysis of Variance of Cross-Validation Estimators of the Generalization Error," *J. Machine Learning Research*, 6, 1127–1168.

[22] Molinaro, A., Simon, R., and Pfeiffer, R. (2005), "Prediction error estimation: a comparison of resampling methods," *Bioinformatics*, 21, 3301–3307.

[23] Polonik, W. and Wang, Z. (2010), "PRIM Analysis," *Journal of Multivariate Analysis*, 101, 525–540.

[24] Simon, R., Radmacher, M., Dobbin, K., and McShane, L. (2003), "Pitfalls in the Use of DNA Microarray Data for Diagnostic and Prognostic Classification," *J. Nat. Cancer Institute*, 95, 14–18.

[25] Simon, R., Subramanian, J., Li, M.-C., and Menezes, S. (2011), "Using cross-validation to evaluate predictive accuracy of survival risk classifiers based on high-dimensional data," *Briefings in Bioinformatics*, 12, 203–214.

[26] Subramanian, J. and Simon, R. (2010), "Gene expression-based prognostic signatures in lung cancer: ready for clinical use?" *J. Natl. Cancer Inst.*, 102, 464474.

[27] Varma, S. and Simon, R. (2006), "Bias in error estimation when using cross-validation for model selection," *BMC bioinformatics*, 7, 91–99.

[28] Wang, P., Kim, Y., Pollack, J., and Tibshirani, R. (2004), "Boosted PRIM with Application to Searching for Oncogenic Pathway of Lung Cancer," in *Computational Systems Bioinformatics Conference, International IEEE Computer Society*, IEEE Computer Society, pp. 604–609.

[29] Wu, L. and Chipman, H. (2003), "Bayesian Model-Assisted PRIM Algorithm," Tech. rep., Departments of Statistics and Actuarial Science, University of Waterloo.