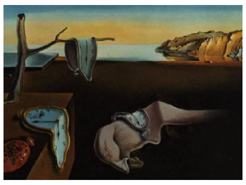

Journal of Statistical Software

May 2021, Volume 98, Issue 5. doi: 10.18637/jss.v098.i05

Subgroup Identification Using the personalized Package

Jared D. Huling
University of Minnesota

Menggang Yu
University of Wisconsin-Madison

Abstract

A plethora of disparate statistical methods have been proposed for subgroup identification to help tailor treatment decisions for patients. However a majority of them do not have corresponding R packages and the few that do pertain to particular statistical methods or provide little means of evaluating whether meaningful subgroups have been found. Recently, the work of Chen, Tian, Cai, and Yu (2017) unified many of these subgroup identification methods into one general, consistent framework. The goal of the **personalized** package is to provide a corresponding unified software framework for subgroup identification analyses that provides not only estimation of subgroups, but evaluation of treatment effects within estimated subgroups. The **personalized** package allows for a variety of subgroup identification methods for many types of outcomes commonly encountered in medical settings. The package is built to incorporate the entire subgroup identification analysis pipeline including propensity score diagnostics, subgroup estimation, analysis of the treatment effects within subgroups, and evaluation of identified subgroups. In this framework, different methods can be accessed with little change in the analysis code. Similarly, new methods can easily be incorporated into the package. Besides familiar statistical models, the package also allows flexible machine learning tools to be leveraged in subgroup identification. Further estimation improvements can be obtained via efficiency augmentation.

Keywords: subgroup identification, heterogeneity of treatment effect, interaction modeling, inverse weighting, individualized treatment rules, precision medicine.

1. Introduction

Many studies of medical interventions, especially clinical trials, often focus on population average treatment effects. However, it is widely recognized that the effects of treatments can have substantial differences across a population. With the increasing interest to improve the efficacy and effectiveness of health care, there has been a significant effort in the statistics community to develop methodology for optimal allocation of treatments to patients. Optimal treatment allocation can be thought of as a subgroup identification task, where subgroups are determined based on the heterogeneity of treatment effect. Heterogeneity of treatment effect can be characterized by the interaction of the treatment with patient characteristics. Thus, the goal in subgroup identification is to characterize and estimate these interactions in order to construct an optimal mapping from patient characteristics to a treatment assignment. This mapping is called an individualized treatment rule (ITR). An optimal ITR is one that, when enacted on a population, results in the largest expected patient outcome, assuming without loss of generality that larger outcomes are preferred. The overall patient outcome is impacted by both the main effects of patient characteristics and the treatment-covariate interactions and thus many approaches, such as Qian and Murphy (2011), focus on modeling this full relationship to estimate ITRs. In their work, Qian and Murphy (2011) show robustness properties to model misspecification under certain conditions. Many recent works have instead focused on methods which do not require correct specification of the entire relationship between patient characteristics, treatment, and outcome, but only the parts relevant to the optimal ITR and are thus often more robust to modeling choices. Regardless of the general modeling approach, a vast majority of methods for ITR estimation do not have corresponding R packages and those that do often pertain to particular statistical methods for optimal ITR estimation (Huang, Sun, Chatterjee, and Trow 2017; Riviere 2021; Dusseldorp, Doove, Van de Put, Mechelen, and Claramunt Gonzalez 2020; der Elst, Alonso, and Molenberghs 2020; Holloway, Laber, Linn, Zhang, Davidian, and Tsiatis 2020; Egami, Ratkovic, and Imai 2019). In addition, there has been much focus on estimation of subgroups based on patient characteristics, yet not enough emphasis on evaluation of the treatment effects within the resulting estimated subgroups, which is an equally important but challenging aspect of any subgroup analysis.

Chen *et al.* (2017) revealed that a wide range of existing statistical methods for optimal ITR estimation fall under the umbrella of a unified estimation framework. This unified framework focuses on the estimation of treatment scores, which rank patients based on their individualized treatment effect. The scoring system encompasses optimal ITR estimation in the sense that a threshold for the treatment score can be used as a treatment assignment mechanism. The **personalized** package (Huling 2021) aims to be a versatile tool in the R statistical language (R Core Team 2021) for optimal ITR estimation and treatment scoring corresponding to the framework of Chen *et al.* (2017). Further, two valid approaches for estimation of treatment effects within the estimated subgroups are provided and can be used straightforwardly with any of the available methods for estimation of treatment scores, enabling validation of fitted subgroup identification models. The **personalized** package offers an entire subgroup analysis workflow with intuitive and easy-to-use structure. Thus, a wide range of subgroup identification methods can be accessed with little change in the analysis workflow of the user. Furthermore the subgroup identification framework allows the practitioner to conduct a subgroup identification analysis using familiar statistical modeling concepts. The package is designed to accommodate a wide range of subgroup identification and treatment decision-making analyses.

The features of the **personalized** package include:

1. A wide range of loss function-based subgroup identification methods.
2. Modeling options for continuous, binary, count, and time-to-event outcomes.

3. Accommodation of observational studies via either propensity score-based analysis or matching.
4. Handling of both binary and multiple treatment scenarios.
5. Efficiency improvements through loss augmentation.
6. Evaluation of estimated subgroups with correction for overfitting.
7. Options for utilizing custom loss functions.

The package is available on the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=personalized` in addition to a development version available as a GitHub repository at `https://github.com/jaredhuling/personalized`.

In Section 2 we provide background on the methodological underpinnings of package **personalized**, followed by a detailed description of the package itself in Section 3. In Section 4 we evaluate the various methods offered in the **personalized** package in a numerical comparison with several methods from other packages. Finally, in Section 5 we demonstrate the use of the package on a subgroup identification analysis of the National Supported Work Study (LaLonde 1986) and conclude with some discussion in Section 6.

2. Subgroup identification framework

2.1. Modeling setup and notation

Individualized treatment effects, benefit scores, and individualized treatment rules

In this section we provide a formal overview of the subgroup identification framework of Chen *et al.* (2017). We first consider binary treatments and then provide extensions to multi-category treatments in a later section. Let the treatment assignment be denoted as $T \in \mathcal{T} = \{-1, 1\}$, where $T = 1$ indicates that a patient received the treatment, and $T = -1$ indicates a patient received the control. We also observe the patient outcome Y, where larger values are assumed to be preferable without loss of generality. We further observe a length p vector of patient covariate information $\mathbf{X} \in \mathcal{X}$. Note that the first element of $\mathbf{X}$ is an intercept term. The covariates may modify the effect of T on Y, resulting in treatment effect heterogeneity. Relating the above to observable quantities, we observe data from n patients $\{(Y_i, T_i, \mathbf{X}_i), i = 1, \ldots, n\}$ consisting of n independent, identically distributed copies of $(Y, T, \mathbf{X})$. In identifying subgroups of patients who may benefit from T differently, we are often interested in estimating the contrast function

$$\Delta(\mathbf{X}) \equiv \mathsf{E}(Y|T = 1, \mathbf{X}) - \mathsf{E}(Y|T = -1, \mathbf{X}). \tag{1}$$

Note that (1) involves a difference of means. There can be cases where a ratio,

$$\Gamma(\mathbf{X}) \equiv \mathsf{E}(Y|T = 1, \mathbf{X})/\mathsf{E}(Y|T = -1, \mathbf{X}), \tag{2}$$

may instead be a more interpretable or relevant estimand, especially for positive Y. Treatment effect heterogeneity is clearly reflected only through either $\Delta(\mathbf{X})$ or $\Gamma(\mathbf{X})$. To see how these quantities relate to the full regression model, note that a completely unspecified regression model $\mathsf{E}(Y|T,\mathbf{X})$ can be expressed in terms of main covariate effects and interactions of the covariates and treatment status:

$$\begin{aligned}\mathsf{E}(Y|T,\mathbf{X}) &= I(T=1)\cdot\mathsf{E}(Y|T=1,\mathbf{X}) + I(T=-1)\cdot\mathsf{E}(Y|T=-1,\mathbf{X})\\ &\equiv g(\mathbf{X}) + T\cdot\Delta(\mathbf{X})/2,\end{aligned}$$

where $I(\cdot)$ is an indicator function and $g(\mathbf{X}) \equiv \frac{1}{2}[\mathsf{E}(Y|T=1,\mathbf{X})+\mathsf{E}(Y|T=-1,\mathbf{X})]$ represents the covariate main effects. Regardless of the form of $\mathsf{E}(Y|T,\mathbf{X})$, the only components that guide which patients benefit from a treatment are the treatment-covariate interactions, $\Delta(\mathbf{X})$. A similar re-expression of $\mathsf{E}(Y|T,\mathbf{X})$ can be shown in terms of $\Gamma(\mathbf{X})$ for positive Y.

Not all subgroup identification methods target $\Delta(\mathbf{X})$ or $\Gamma(\mathbf{X})$ directly, but may rather target some useful transformation of them. To formalize this notion, we define a *benefit score* to be any mapping $f(\mathbf{X})$ that possesses the following two properties: i) it reflects the degree to which individual patients "benefit" from a treatment, i.e., is monotone in the treatment effect $\Delta(\mathbf{X})$, $\Gamma(\mathbf{X})$, or otherwise; ii) it has a meaningful, known cutpoint value c such that for a given level of covariates $\mathbf{x}$, $f(\mathbf{x}) > c$ implies that the treatment is more effective than the control (e.g., $\Delta(\mathbf{X}) > 0$) and $f(\mathbf{x}) \leq c$ implies the control is more effective than the treatment (e.g., $\Delta(\mathbf{X}) \leq 0$). Clearly $\Delta(\mathbf{X})$ is itself a benefit score as it reflects how much a patient is expected to benefit from a treatment in terms of his or her outcome. For a patient with $\mathbf{X} = \mathbf{x}$, $\Delta(\mathbf{x}) > 0$ indicates that the treatment is "better" in terms of the expected outcome whereas $\Delta(\mathbf{x}) < 0$ indicates that the control is better. Hence, estimation of $\Delta(\mathbf{X})$ or its sign allows recommending different subgroups of patients to different treatments in an optimal manner. By definition, $\Delta(\mathbf{X})$ can also be used to rank patients by the magnitude of treatment effect. $\Gamma(\mathbf{X})$ is clearly also a benefit score where $\Gamma(\mathbf{x}) > 1$ indicates that the treatment is better in terms of the expected outcome and $\Gamma(\mathbf{x}) \leq 1$ indicates the reverse. It is easily seen that the use of either $\Delta(\mathbf{x})$ or $\Gamma(\mathbf{x})$ should lead to similar subgroups.

In Section 2.2 we will introduce benefit score estimators $\hat{f}(\mathbf{X})$ which can either identify the patients for whom a treatment is better than a control or rank patients by the degree of "benefit" a treatment has. These estimators are not always estimators of $\Delta(\mathbf{X})$ directly, but often monotone transformations of $\hat{f}(\mathbf{X})$ will yield estimators of $\Delta(\mathbf{X})$, i.e., $\widehat{\Delta}(\mathbf{X}) = h(\hat{f}(\mathbf{X}))$ for some monotone $h(\cdot)$.

Another quantity of interest is an ITR, which is a map from patient characteristics to treatment decisions $d(\mathbf{X}) : \mathcal{X} \mapsto \mathcal{T}$. An optimal ITR maximizes the value function $V(d) = \mathsf{E}^d(Y) = \int Y \mathrm{d}P^d$, where P^d is the distribution of $(Y,T,\mathbf{X})$ given $T = d(\mathbf{X})$. Essentially, optimal ITRs make treatment decisions for patients in a manner such that the average outcomes across the population are maximized. Both $\Delta(\mathbf{X})$ and $\Gamma(\mathbf{X})$ can be used to construct optimal ITRs. In particular, $\text{sign}\{\Delta(\mathbf{X})\}$ and $\text{sign}\{\Gamma(\mathbf{X}) - 1\}$ are optimal ITRs.

Assumptions for causal interpretations

To deal with non-randomized treatment assignment in observational studies, as in Chen *et al.* (2017), we adopt the notation from the potential outcome framework of Rubin (2005). Let $Y^{(1)}$ and $Y^{(-1)}$ be the potential outcomes if the patient receives $T = 1$ and $T = -1$, respectively. In reality only one of the potential outcomes can be observed for each individual. Formally, this statement can be enforced by the relation $Y = I(T=1)Y^{(1)} + I(T=-1)Y^{(-1)}$, where $I(\cdot)$ is the indicator function, under the stable unit treatment value assumption (SUTVA; Rubin 2005). In essence, SUTVA requires the potential outcome of a unit when exposed to a treatment will be the same no matter what mechanism is used to assign the treatment. We also assume "strong ignorability" (Rosenbaum and Rubin 1983; Rubin 2005), that is, $T \perp\!\!\!\perp (Y^{(1)}, Y^{(-1)}) \,|\, \mathbf{X}$. Violations of SUTVA can occur when there are spillover effects from treated units to other units, however in this paper we always assume SUTVA holds. We assume that the treatment assignment mechanism is either known, as is the case in randomized controlled trials, or is unknown and can be estimated, as is the case when there are no unmeasured confounders. In other words, $\pi(\mathbf{X}) = \mathsf{P}(T = 1|\mathbf{X})$ is either a known function or can be consistently estimated via regression modeling. Further, a "positivity" assumption that all patients have a chance of receiving the treatment, i.e., $0 < \pi(\mathbf{x}) < 1$ for all $\mathbf{x} \in \mathcal{X}$, is required. Under these assumptions, $\Delta(\mathbf{X}) = \mathsf{E}(Y^{(1)}|\mathbf{X}) - \mathsf{E}(Y^{(-1)}|\mathbf{X})$ and $\Gamma(\mathbf{X}) = \mathsf{E}(Y^{(1)}|\mathbf{X})/\mathsf{E}(Y^{(-1)}|\mathbf{X})$ are treatment effects conditional on patient characteristics. Note that in the potential outcome notation, the value function is $V(d) = \mathsf{E}[Y^{(d)}]$. Many matching strategies (Imbens and Rubin 2015) can also be used instead of direct modeling of $\pi(\mathbf{X}) = \mathsf{P}(T = 1|\mathbf{X})$. However, note that under matching, the targeted estimand can depend on the matching mechanism. For example, if matching is based on the treated subjects, then the estimand is the treatment effect on the treated conditional on patient characteristics, i.e., $\Delta_1(\mathbf{X}) = \mathsf{E}(Y^{(1)}|\mathbf{X}, T = 1) - \mathsf{E}(Y^{(-1)}|\mathbf{X}, T = 1)$ or $\Gamma_1(\mathbf{X}) = \mathsf{E}(Y^{(1)}|\mathbf{X}, T = 1)/\mathsf{E}(Y^{(-1)}|\mathbf{X}, T = 1)$.

2.2. Benefit score estimators and their properties

Subgroup identification and benefit score estimation via loss functions

The framework of Chen *et al.* (2017) covers two classes of benefit score estimators. The two methods, called the weighting method and the advantage-learning (A-learning) method, are both quite general approaches for estimating $\Delta(\mathbf{X})$ or $\Gamma(\mathbf{X})$ (or transformations of $\Delta(\mathbf{X})$ or $\Gamma(\mathbf{X})$) via loss functions. Both the weighting and the A-learning methods do not require specification of the full outcome regression model and focus on direct estimation of $\Delta(\mathbf{X})$, $\Gamma(\mathbf{X})$, or transformations thereof. As we will explore in later sections, outcome regression models can, however, be incorporated into both the weighting and A-learning methods in order to improve efficiency. A major benefit of both the weighting and A-learning methods is that even when full outcome regression models are utilized, misspecification of the full outcome regression model does not impact the validity of the resulting estimators.

Consider a convex loss function $\mathbf{M}(y, v)$ used for the purpose of estimating benefit scores. A useful example is the squared error loss, $\mathbf{M}(y, v) = (y - v)^2$. In their original work, Chen *et al.* (2017) require $\mathbf{M}(y, v)$ to meet the following conditions i) $\mathbf{M}_v(y, v) = \partial\mathbf{M}(y, v)/\partial v$ is increasing in v for every fixed y and ii) $\mathbf{M}_v(y, 0)$ is monotone in y. These requirements are sufficient for Fisher consistent subgroup identification, however, they are not necessary. Conditions i) and ii) can be relaxed to incorporate a wider class of losses such as the hinge loss $\mathbf{M}(y, v) = y \max(1 - v, 0)$. In Section 2.4, we point out that the conditions specified by Chen *et al.* (2017) on $\mathbf{M}$ for the multi-category treatment setting can also be relaxed.

Weighting method

The first estimation method is called the weighting method. Given a sample of n patients, the weighting method estimates $\Delta(\mathbf{X})$ or $\Gamma(\mathbf{X})$ (or transformations thereof) by minimizing the following objective function with respect to $f(\mathbf{X})$:

$$L_W(f) = \frac{1}{n}\sum_{i=1}^{n} \frac{\mathbf{M}(Y_i, T_i \times f(\mathbf{X}_i))}{T_i\pi(\mathbf{X}_i) + (1 - T_i)/2}, \tag{3}$$

where W indicates the weighting method and $\pi(\mathbf{x}) = \mathsf{P}(T = 1|\mathbf{X} = \mathbf{x})$ is the propensity score function. The weighting estimator is then $\hat{f}_W = \text{argmin}_f L_W(f)$. The corresponding population level weighting estimator is the minimizer of $\ell_W(f) = E[\ell_W(f, \mathbf{x})]$, where

$$\ell_W(f, \mathbf{x}) = \mathsf{E}\left[\frac{\mathbf{M}(Y, T \times f(\mathbf{x}))}{T\pi(\mathbf{x}) + (1 - T)/2}|\mathbf{X} = \mathbf{x}\right], \tag{4}$$

with respect to f, where W again indicates the weighting method. The weighting method is valid without specification of the full outcome regression model, as the inverse weights result in the interactions $T \times f(\mathbf{X})$ being uncorrelated with the main effects $g(\mathbf{X})$. The estimated benefit score under the weighting method, $\hat{f}_W$, can be used to estimate $\Delta(\mathbf{X})$ under many different loss functions. See Table 1 for examples.

A-learning method

The A-learning estimator involves minimizing

$$L_A(f) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{M}(Y_i, \{(T_i + 1)/2 - \pi(\mathbf{X}_i)\} \times f(\mathbf{X}_i)), \tag{5}$$

where A indicates the A-learning method and $(T_i + 1)/2 = I(T_i = 1)$. The A-learning estimator is then $\hat{f}_A = \text{argmin}_f L_A(f)$. The A-learning method works without specification of the full regression model, because the centered interaction $\{(T + 1)/2 - \pi(\mathbf{X})\} \times f(\mathbf{X})$ is uncorrelated with, and in fact orthogonal to, the main effects $g(\mathbf{X})$. This property follows from the fact that $\mathsf{E}[(T + 1)/2 - \pi(\mathbf{X})|\mathbf{X}]$ is zero. The corresponding population level A-learning estimator is the minimizer of

$$\ell_A(f, \mathbf{x}) = \mathsf{E}\left[\mathbf{M}(Y, \{(T + 1)/2 - \pi(\mathbf{x})\} \times f(\mathbf{x}))|\mathbf{X} = \mathbf{x}\right] \tag{6}$$

with respect to f, where again A indicates the A-learning method and $(T + 1)/2 = I(T = 1)$.

Benefit score properties and estimands

Although $\hat{f}_W(\cdot)$ and $\hat{f}_A(\cdot)$ are not always themselves estimates of $\Delta(\cdot)$, the zero point for $\hat{f}_W(\cdot)$ and $\hat{f}_A(\cdot)$ is *always* meaningful and can be used as a threshold for determining subgroups. For example, assuming that larger outcomes are preferred, all patients with covariates $\mathbf{x}$ such that $f_W(\mathbf{x}) > 0$ should have better outcomes under the treatment than under the control on average. More formally, denote the population estimators to be

$$f_{W0}(\mathbf{x}) = \text{argmin}_f \mathsf{E}\{\ell_W(f, \mathbf{x})\} \text{ and } f_{A0}(\mathbf{x}) = \text{argmin}_f \mathsf{E}\{\ell_A(f, \mathbf{x})\}.$$

Under both the weighting and A-learning methods and conditions i) and ii) of M from above, for patients with a negative benefit score score ($f_{A0}(\mathbf{x}) < 0$ or $f_{W0}(\mathbf{x}) < 0$), we have $\mathsf{E}\{\mathbf{U}(Y^{(1)})|\mathbf{X} = \mathbf{x}\} > \mathsf{E}\{\mathbf{U}(Y^{(-1)})|\mathbf{X} = \mathbf{x}\}$ where $\mathbf{U}(y) = \partial\mathbf{M}(y, v)/\partial v|_{v=0}$ and for those with a positive benefit score, we have $\mathsf{E}\{\mathbf{U}(Y^{(1)})|\mathbf{X} = \mathbf{x}\} < \mathsf{E}\{\mathbf{U}(Y^{(-1)})|\mathbf{X} = \mathbf{x}\}$. Thus,

$d_{W0}(\mathbf{x}) = \text{sign}(f_{W0}(\mathbf{x}))$ and $d_{A0}(\mathbf{x}) = \text{sign}(f_{A0}(\mathbf{x}))$ are optimal decision rules for mapping patient characteristics $\mathbf{X}$ to treatment decision T. Note that treatment assignment decisions here, while based on the overall cutoff value of 0, are determined by the individual treatment effects. It was shown in Chen *et al.* (2017) that the estimates resulting from both the weighting and A-learning methods result in Fisher-consistent treatment decision rules under a wide class of types of outcomes and $\mathbf{M}$ functions. Hence, the estimated benefit scores can be used to optimally assign patients to treatment groups. For non-differentiable losses such as the hinge loss, similar arguments can be made.

Furthermore, the benefit scores themselves can reflect the magnitude of the individual treatment effect and thus can be used for ranking patients by how effective the treatment is. For example if we use $\mathbf{M}(y, v) = (y - v)^2$, then

$$2f_{W0}(\mathbf{x}) = \mathsf{E}(Y^{(1)}|\mathbf{X} = \mathbf{x}) - \mathsf{E}(Y^{(-1)}|\mathbf{X} = \mathbf{x}) = \Delta(\mathbf{x})$$

and

$$f_{A0}(\mathbf{x}) = \mathsf{E}(Y^{(1)}|\mathbf{X} = \mathbf{x}) - \mathsf{E}(Y^{(-1)}|\mathbf{X} = \mathbf{x}) = \Delta(\mathbf{x}).$$

Other choices of $\mathbf{M}(y, v)$ lead to different interpretations. See Table 1 for more examples of the relationship between f_{W0}, f_{A0} and $\Delta(\mathbf{X})$ or $\Gamma(\mathbf{X})$. As pointed out in Chen *et al.* (2017), similar to using surrogate loss functions in a classification setting (Bartlett, Jordan, and McAuliffe 2006), the final form of the solution, f_{W0} or f_{A0}, depends on the choice of the loss functions. However, not all choices of $\mathbf{M}(y, v)$ lead to such direct interpretation. For example, the hinge loss $\mathbf{M}(y, v) = y \max(0, 1 - v)$ does not seem to have a direct link with $\Delta(\mathbf{X})$, though the zero point of both f_{W0} and f_{A0} under the hinge loss is still meaningful. The **personalized** package offers estimation under more losses than are listed in Table 1, however the additional losses lead to less interpretable estimates. A listing of all losses implemented in the **personalized** package is available in Table 2.

In scenarios where there are limited resources to allocate treatments, it may be of interest to find a smaller subgroup of patients to recommend the treatment than the subgroup resulting from patients with $f_{W0} > 0$. Since f_{W0} and f_{A0} rank patients by magnitude of treatment effect under most losses, and thus using $f_{W0} > c$ with $c > 0$ yields a smaller subgroup with larger treatment effect than $f_{W0} > 0$. Thus, given limited resources for treatment allocation, using $f_{W0} > c$ can be useful to find a small subgroup of patients for whom the treatment is highly beneficial.

Loss function choices and relationship with other methods

Although many of the loss functions in Table 1 are related to negative log-likelihoods from specific models and distributions, in general there is no distributional requirement of outcomes for specific choices of losses, except for the loss corresponding to the Cox proportional hazards model. Thus, if the outcome of interest is a count outcome, it is valid to use losses other than $\mathbf{M}(y, v) = -[yv - \exp(v)]$, such as the squared loss, hinge loss, or others. Similarly, it is valid to use losses other than the logistic loss, $\mathbf{M}(y, v) = -[yv - \log(1 + \exp\{-v\})]$.

Each combination of the A-learning or weighting methods with a valid loss function results in a different estimator, allowing for a high degree of versatility in estimation. For example, $\mathbf{M}(y, v) = y \log\{1 + \exp(-v)\}$ corresponds to the method developed in Xu, Yu, Zhao, Li, Wang, and Shao (2015), $\mathbf{M}(y, v) = y \max(1 - v, 0)$ corresponds to the outcome weighted learning (OWL) method of Zhao, Zeng, Rush, and Kosorok (2012), under a randomized

$\mathbf{M}(y, v)$	Estimand	Weighting	A-learning
$(y-v)^2$	$\Delta(\mathbf{X})$	$2f_{W0}(\mathbf{X})$	$f_{A0}(\mathbf{X})$
$-[yv - \exp(v)]$	$\Delta(\mathbf{X})$	$\exp\{f_{W0}(\mathbf{X})\} - \exp\{-f_{W0}(\mathbf{X})\}$	$\exp\{(1-\pi(\mathbf{X}))f_{A0}(\mathbf{X})\} - \exp\{-\pi(\mathbf{X})f_{A0}(\mathbf{X})\}$
$-[yv - \log(1+\exp\{-v\})]$	$\Delta(\mathbf{X})$	$\frac{\exp\{f_{W0}(\mathbf{X})\}-1}{\exp\{f_{W0}(\mathbf{X})\}+1}$	$\frac{(\exp\{f_{A0}(\mathbf{X})\}-1)}{(\exp\{\pi(\mathbf{X})f_{A0}(\mathbf{X})\}+1)} \times \frac{1}{1+\exp\{(1-\pi(\mathbf{X}))f_{A0}(\mathbf{X})\}}$
$y \log(1+\exp\{-v\})$	$\Gamma(\mathbf{X})$	$\exp\{f_{W0}(\mathbf{X})\}$	$\frac{1+\exp\{(1-\pi(\mathbf{X}))f_{A0}(\mathbf{X})\}}{1+\exp\{-\pi(\mathbf{X})f_{A0}(\mathbf{X})\}}$
$-\left\{\int_0^\tau (v - \log[\mathsf{E}\{e^v I(X \geq u)\}])\,\mathrm{d}N(u)\right\}$	$\Gamma^*_M(\mathbf{X})^\dagger$	$\exp\{-f_{W0}(\mathbf{X})\}$	$\exp\{-f_{A0}(\mathbf{X})\}$

†Censoring rates are assumed to be equal within treatment arms.

Table 1: The last loss above is for survival outcomes with $y = (X, \delta) = \{\widetilde{X} \wedge C, I(\widetilde{X} \leq t)\}$, $\widetilde{X}$ is the survival time, C is the censoring time, $N(t) = I(\widetilde{X} \leq t)\delta$, and τ is a fixed point such that $\mathsf{P}(X \geq \tau) > 0$. The term $\Gamma^*_M(\mathbf{X})$ for $M \in \{W, A\}$ above is defined as $\frac{\mathsf{E}[\Lambda^*_M(Y^\dagger)|T=1,\mathbf{X}]}{\mathsf{E}[\Lambda^*_M(Y^\dagger)|T=-1,\mathbf{X}]}$, where $\Lambda^*_W(t)$ is a monotone increasing function described in the Appendix of Tian *et al.* (2014) and $\Lambda^*_A(t)$ is quite similar to $\Lambda^*_W(t)$. $\Gamma^*_W(\mathbf{X})$ corresponds to the estimand of the weighting method and $\Gamma^*_A(\mathbf{X})$ corresponds to the estimand of the A-learning method. Under randomization into treatment and control groups with equal probability, the forms above for $\Gamma(\mathbf{X})$ or $\Delta(\mathbf{X})$ for the A-learning method simplify dramatically. For example, under equal randomization and $\mathbf{M}(y, v) = y\log(1+\exp\{-v\})$, $\Gamma(\mathbf{X}) = \exp\{f_{A0}(\mathbf{X})/2\}$.

clinical trial setting with treatments assigned with equal probability, both the A-learning and weighting methods with $\mathbf{M}(y, v) = (y-v)^2$ reduce to the modified covariate method of Tian *et al.* (2014) for continuous responses, the A-learning method with $\mathbf{M}(y, v) = (y-v)^2$ corresponds to the approach of Lu, Zhang, and Zeng (2013) and Ciarleglio, Petkova, Ogden, and Tarpey (2015), among others. Using the A-learning method with the squared error loss and loss augmentation (described below in Section 2.5) is equivalent, after accounting for any variable selection penalties, to the estimation method utilized in Zhao, Small, and Ertefaie (2017), Shi, Song, and Lu (2016), Shi, Fan, Song, Lu *et al.* (2018), and the method behind the de-sparsified estimator of Jeng, Lu, Peng *et al.* (2018).

Modeling choices for the benefit score

Modeling choices must be made for the form of f_W and f_A. One can use a simple form of f such as a linear combination of the covariates, i.e., $f(\mathbf{X}) = \mathbf{X}^\top\boldsymbol{\beta}$. Hence $\hat{f}(X) = \mathbf{X}^\top\hat{\boldsymbol{\beta}}$. Such a choice leads to interpretable models. For most loss functions, if the effect β_j of variable j is positive, then increased values of variable j lead to an increase in treatment benefit and negative effects lead to decreased benefit. Beyond linear forms, regression trees, smoothing splines, or other nonparametric and flexible approaches may be used for f_W or f_A.

2.3. Loss function example and implementation details

One of the key benefits of the framework of Chen *et al.* (2017) is the relative ease of implementation. Many combinations of method (weighting or A-learning) and loss function can be computed by existing regression software. Either (3) or (5) can be minimized for a given loss by providing existing software a modified covariate $\widetilde{\boldsymbol{X}}$, provided the existing software can accept observation weights.

To see how this is accomplished, consider the familiar example of the squared error loss function $\mathbf{M}(y, v) = (y - v)^2$. Under this loss and the assumption that $\Delta(\mathbf{X}) = \mathbf{X}^\top \boldsymbol{\beta}$, we can minimize (3) using existing software. First, denote the $n \times p$ design matrix of patient covariate information to be $\boldsymbol{X}$. Denote a modified design matrix $\widetilde{\boldsymbol{X}}$ to be $\text{diag}(\boldsymbol{T})\boldsymbol{X}$, where $\boldsymbol{T} = (T_1, \ldots, T_n)^\top$ and $\text{diag}(\boldsymbol{T})$ is the diagonal matrix with diagonal elements as the elements of the vector $\boldsymbol{T}$. Then the minimizer of (3) is simply the weighted least squares estimator

$$(\widetilde{\boldsymbol{X}}^\top \text{diag}(\boldsymbol{W})\widetilde{\boldsymbol{X}})^{-1}\widetilde{\boldsymbol{X}}^\top \text{diag}(\boldsymbol{W})\boldsymbol{Y},$$

where $\boldsymbol{W}$ is a vector of weights with the i-th element as $1/(T_i\pi(\mathbf{x}_i) + (1 - T_i)/2)$ and $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\top$. If $\widetilde{\boldsymbol{X}}$ is high dimensional and variable selection is desired, $\widetilde{\boldsymbol{X}}$ and $\boldsymbol{Y}$ along with a vector of observation weights can be supplied to existing software, such as the **glmnet** R package (Friedman, Hastie, Tibshirani, Narasimhan, Tay, and Simon 2021). More details on how this is handled in the **personalized** package are provided in Section 3.

More generally, existing software can be used to minimize (3) and (5) by appropriate construction of weights and modified design matrices. The modified design matrix for the weighting method is defined as in the example above, and the modified design matrix for the A-learning method is defined as $\widetilde{\boldsymbol{X}} = \text{diag}((\boldsymbol{T}+1)/2 - \boldsymbol{\pi}(\boldsymbol{X}))\boldsymbol{X}$ where $\boldsymbol{\pi}(\boldsymbol{X})$ is a vector with i-th element equal to $\pi(\mathbf{x}_i)$.

2.4. Extension to multi-category treatments

Often, more than two treatments are available for patients and the researcher may wish to understand which of all treatment options are the best for which patients. Extending the above methodology to multi-category treatment results in added complications, and in particular there is no straightforward extension of the A-learning method for multiple treatment settings. In the supplementary materials of Chen *et al.* (2017), the weighting method was extended to estimate a benefit score corresponding to each level of a treatment subject to a sum-to-zero constraint for identifiability. In particular, we are interested in estimating (the sign of)

$$\Delta_{kl}(\mathbf{x}) \equiv \mathsf{E}(Y|T = k, \mathbf{X} = \mathbf{x}) - \mathsf{E}(Y|T = l, \mathbf{X} = \mathbf{x}). \tag{7}$$

If $\Delta_{kl}(\mathbf{x}) > 0$, then treatment k is preferable to treatment l for a patient with $\mathbf{X} = \mathbf{x}$. For each patient, evaluation of all pairwise comparisons of the $\Delta_{kl}(\mathbf{x})$ indicates which treatment leads to the largest expected outcome. The weighting estimators of the benefit scores are the minimizers of the following loss function:

$$L_W(f_1, \ldots, f_K) = \frac{1}{n}\sum_{i=1}^{n} \frac{\mathbf{M}(Y_i, \sum_{k=1}^K I(T_i = k) \times f_k(\mathbf{X}_i))}{\mathsf{P}(T = T_i|\mathbf{X} = \mathbf{X}_i)} \tag{8}$$

subject to $\sum_{k=1}^K f_k(\mathbf{X}_i) = 0$. Clearly when $K = 2$, this loss function is equivalent to (3).

Estimation of the benefit scores in this model is still challenging without added modeling assumptions, as enforcing $\sum_{k=1}^K f_k(\mathbf{X}_i) = 0$ may not always be feasible using existing estimation routines. However, if each $\Delta_{kl}(\mathbf{X})$ has a linear form, i.e., $\Delta_{kl}(\mathbf{X}) = \mathbf{X}^\top \boldsymbol{\beta}_k$ where l represents a reference treatment group, estimation can then easily be fit into the same computational framework as for the simpler two treatment case by constructing an appropriate design matrix. Thus, for multiple treatments the **personalized** package is restricted to linear estimators of the benefit scores. For instructive purposes, consider a scenario with three treatment options, A, B, and C. Let $\boldsymbol{X} = (\boldsymbol{X}_A^\top, \boldsymbol{X}_B^\top, \boldsymbol{X}_C^\top)^\top$ be the design matrix for all patients, where each $\boldsymbol{X}_k^\top$ is the sub-design matrix of patients who received treatment k. Under $\Delta_{kl}(\mathbf{X}) = \mathbf{X}^\top \boldsymbol{\beta}_k$ with l as the reference treatment, we can construct a new design matrix which can then be provided to existing estimation routines in order to minimize (8). With treatment C as the reference treatment, the design matrix is constructed as

$$\widetilde{\boldsymbol{X}} = \mathrm{diag}(\boldsymbol{J}) \begin{pmatrix} \boldsymbol{X}_A & \mathbf{0} \\ \mathbf{0} & \boldsymbol{X}_B \\ \boldsymbol{X}_C & \boldsymbol{X}_C \end{pmatrix},$$

where the i-th element of $\boldsymbol{J}$ is $2I(T_i \neq C) - 1$ and the weight vector $\boldsymbol{W}$ is constructed with the i-th element set to $1/\mathsf{P}(T = T_i|\mathbf{X} = \mathbf{X}_i)$. Furthermore denote $\widetilde{\boldsymbol{\beta}} = (\boldsymbol{\beta}_A^\top, \boldsymbol{\beta}_B^\top)^\top$. Hence $\widetilde{\boldsymbol{X}}^\top \widetilde{\boldsymbol{\beta}} = \boldsymbol{X}_A^\top \boldsymbol{\beta}_A + \boldsymbol{X}_B^\top \boldsymbol{\beta}_B - \boldsymbol{X}_C^\top(\boldsymbol{\beta}_A + \boldsymbol{\beta}_B)$, and thus the sum-to-zero constraints on the benefit scores hold by construction.

The conditions specified for the loss functions in the supplementary material of Chen *et al.* (2017) are too restrictive. In general, we find that all Fisher-consistent margin based classification loss functions (Bartlett *et al.* 2006; Tewari and Bartlett 2007; Zou, Zhu, and Hastie 2008; Tewari and Bartlett 2007) can be adopted for the weighting method in the multi-category treatment setting. The sum-to-zero constraint may be elegantly solved by mimicking the angle-based reformulation proposed by Zhang and Liu (2014) in the classification setting. However this approach has not yet been adopted in the **personalized** package.

2.5. Efficiency improvement via loss function augmentation

As mentioned in previous sections, the weighting and A-learning approaches do not require the specification of a full outcome regression model to consistently recover optimal subgroups. However, gains in efficiency can be made through augmentation of the loss function. Loss augmentation involves positing and fitting a full outcome regression model and using the predictions from this model to construct a modified loss function. This approach has a few benefits: First, if the full outcome regression model is indeed correctly specified, then the resulting estimator will be efficient and second, if the outcome regression model is incorrectly specified, it does not impact the Fisher consistency of the estimated subgroups. Further, in practice even when the outcome regression model is incorrect, efficiency gains are often realized.

The basic approach to augmentation is the following:

1. Fit a regression model for the conditional mean $\mathsf{E}[Y|T, \mathbf{X}] = g(\mathbf{X}) + T\Delta(\mathbf{X})$ and create predictions $\widehat{\mathsf{E}}[Y|T, \mathbf{X}]$. When the conditional mean is linked to predictors via a link function, fit a regression model for $h(\mathsf{E}[Y|T, \mathbf{X}]) = g(\mathbf{X}) + T\Delta(\mathbf{X})$, where $h(\cdot)$ is a known link function, and generate predictions on the scale of the linear predictor, $\mathrm{pred}(\mathbf{X}, T) = h(\widehat{\mathsf{E}}[Y|T, \mathbf{X}])$.

2. Create the augmentation function by integrating predictions over both treatment options: $a(\mathbf{X}_i) = \sum_{T \in \mathcal{T}} a_T \text{pred}(\mathbf{X}_i, T)$, where a_T are weights. In practice, a simple average with $a_T = 1/2$ works well.
3. Construct an augmented loss function $\widetilde{\mathbf{M}}(y, v) = \mathbf{M}(y, v) + \mathbf{g}(a(\mathbf{X}), v)$, where $\mathbf{g}(y, v)$ meets the same conditions required of $\mathbf{M}(y, v)$.

The augmented loss function $\widetilde{\mathbf{M}}(y, v)$ does not change the optimality of the resulting decision rules, and thus allows for potential reductions in variance. The most efficient augmentation function under the class of augmentation functions of the multiplicative form $v\mathbf{g}(\mathbf{X})$ was derived in the supplementary material of Chen *et al.* (2017). However, in the **personalized** package, we consider a more limited class of augmentation functions $\mathbf{g}^*(\mathbf{X}) = \mathbf{g}(a(\mathbf{X}))$ that allows for simpler implementation using the functionality for offsets provided in existing software.

2.6. Validating subgroups via subgroup-conditional treatment effects

A subgroup analysis under the framework of Chen *et al.* (2017) involves estimating a set of benefit scores $\hat{f}(\mathbf{X}_i)$ based on $\{(Y_i, T_i, \mathbf{X}_i), i = 1, \ldots, n\}$ and then using the benefit scores to construct subgroups, i.e., a subgroup of patients for whom the treatment is "recommended" via $\hat{f}(\mathbf{X}_i) > 0$ and a subgroup of patients for whom the treatment is not recommended, $\hat{f}(\mathbf{X}_i) \leq 0$. Upon using the data for this purpose, some natural questions to ask are "what is the effect of the treatment among those with $\hat{f}(\mathbf{X}_i) > 0$?", "is the effect of the treatment among those with $\hat{f}(\mathbf{X}_i) > 0$ meaningfully different from zero?", and "what would be the improvement in outcomes over the population of interest if all patients followed the recommendations of $\hat{f}(\mathbf{X}_i)$?". Such questions are nontrivial to answer using the same n samples used to construct the subgroups. The subgroups themselves are conditional on the observed outcomes, thus simply evaluating treatment effects within subgroups will yield biased and overly-optimistic estimates of the subgroup-conditional treatment effects. See Athey and Imbens (2016) and Qiu, Zeng, and Wang (2018) for more discussion.

Denote the decision rule under benefit scores $\hat{f}$ as $d(\mathbf{X}) = \text{sign}(\hat{f}(\mathbf{X}))$. Then the overall benefit of the treatment assignment rule d over the population is

$$\delta(d) = \mathsf{E}(Y^{(d(\mathbf{X}))} | d(\mathbf{X}) = T) - \mathsf{E}(Y^{(-d(\mathbf{X}))} | d(\mathbf{X}) = T).$$

Further define the following subgroup-conditional treatment effects as

$$\begin{aligned}\delta_1(d) &= \mathsf{E}(Y^{(1)} | d(\mathbf{X}) = 1) - \mathsf{E}(Y^{(-1)} | d(\mathbf{X}) = 1) \\ &= \mathsf{E}(Y | T = 1, d(\mathbf{X}) = 1) - \mathsf{E}(Y | T = -1, d(\mathbf{X}) = 1)\end{aligned}$$

and

$$\begin{aligned}\delta_{-1}(d) &= \mathsf{E}(Y^{(-1)} | d(\mathbf{X}) = -1) - \mathsf{E}(Y^{(1)} | d(\mathbf{X}) = -1) \\ &= \mathsf{E}(Y | T = -1, d(\mathbf{X}) = -1) - \mathsf{E}(Y | T = 1, d(\mathbf{X}) = -1).\end{aligned}$$

The quantities $\delta_1(d)$ and $\delta_{-1}(d)$ may also be of interest. Directly calculating empirical versions, $\hat{\delta}(d)$, $\hat{\delta}_1(d)$, and $\hat{\delta}_{-1}(d)$ as defined below, of the above quantities using the following will yield biased estimates. The biased, empirical estimates are

$$\hat{\delta}_t(d) = \frac{\sum_i I\{d(\mathbf{X}_i) = T_i = t\} Y_i}{\sum_i I\{d(\mathbf{X}_i) = T_i = t\}} - \frac{\sum_i I\{d(\mathbf{X}_i) = T_i = -t\} Y_i}{\sum_i I\{d(\mathbf{X}_i) = T_i = -t\}}$$

for $t \in \{1, -1\}$ and

$$\hat{\delta}(d) = \frac{\sum_i I\{d(\mathbf{X}_i) = T_i\}Y_i}{\sum_i I\{d(\mathbf{X}_i) = T_i\}} - \frac{\sum_i I\{d(\mathbf{X}_i) \neq T_i\}Y_i}{\sum_i I\{d(\mathbf{X}_i) \neq T_i\}}.$$

Thus, alternative approaches are needed to estimate $\delta(d)$, $\delta_1(d)$, and $\delta_{-1}(d)$. Another potentially interesting statistic to measure benefit of subgroup recommendations is the C-for-benefit statistic of Van Klaveren, Steyerberg, Serruys, and Kent (2018), however this is not used in the **personalized** package.

Bootstrap bias correction

The first approach used in the **personalized** package to estimate $\delta(d)$, $\delta_1(d)$, and $\delta_{-1}(d)$ is the bootstrap bias correction approach of Harrell, Lee, and Mark (1996). The bootstrap bias correction approach seeks to estimate the bias in the estimates of the subgroup treatment effects that arise from using the same data to estimate these effects as was used to construct the subgroups and then corrects for this bias. This bootstrap bias correction method was introduced in Harrell *et al.* (1996) and later used in Foster, Taylor, and Ruberg (2011) for the purpose of evaluating subgroup effectiveness. For any statistic S, let $S_{\text{full}}(\boldsymbol{X})$ be the statistic estimated with the full training data $\{(Y_i, T_i, \mathbf{X}_i), i = 1, \ldots, n\}$ and evaluated on data $\boldsymbol{X}$ and $S_b(\boldsymbol{X})$ be the statistics estimated using a bootstrap sample $\boldsymbol{X}_b$ (samples from $\{(Y_i, T_i, \mathbf{X}_i), i = 1, \ldots, n\}$) and evaluated on $\boldsymbol{X}$. The general outline of the bootstrap bias correction method is as follows:

- Construct B bootstrap samples of size n with replacement. For the b-th bootstrap sample calculate the statistic $S_b(\boldsymbol{X})$ and $S_b(\boldsymbol{X}_b)$.
- The bootstrap estimate of the amount of bias with regards to the statistic S is

$$\text{bias}_S(\boldsymbol{X}) = \frac{1}{B}\sum_{b=1}^{B}\left[S_b(\boldsymbol{X}_b) - S_b(\boldsymbol{X})\right].$$

- The bias-corrected estimate of the statistic S is then calculated as

$$S_{\text{full}}(\boldsymbol{X}) - \text{bias}_S(\boldsymbol{X}).$$

The term $S_b(\boldsymbol{X}_b)$ involves evaluating statistic S on the same data as was used to construct the underlying estimator. The term $S_b(\boldsymbol{X})$ involves evaluating the S on the original dataset, which acts like an external dataset. Thus, $S_b(\boldsymbol{X}_b) - S_b(\boldsymbol{X})$ mimics the bias that arises from evaluating a statistic on the same dataset that was used to construct the statistic/estimator.

The **personalized** package uses the bootstrap bias correction procedure to estimate $\delta(d)$, $\delta_1(d)$, and $\delta_{-1}(d)$.

Repeated training/testing splitting

The training and testing splitting approach we outline in this section is similar to the sample-splitting scheme of Qiu *et al.* (2018), however their approach is based on a K-fold type procedure, whereas ours is based on repeated splitting of the data. The **personalized** package has functionality for using the training/testing splitting procedure to estimate $\delta(d)$, $\delta_1(d)$, and

$\delta_{-1}(d)$. The procedure involves repeatedly randomly partitioning the data into a training portion and a testing portion. Each replication partitions $\tau \times 100\%$ of the data into the training data and $(1-\tau) \times 100\%$ into the testing data. Using similar notation as for the bootstrap bias correction approach, define for the b-th replication $S_{\text{train},b}(\boldsymbol{X})$ to be statistic S constructed using the b-th training sample and evaluated on data $\boldsymbol{X}$ and $\boldsymbol{X}_{\text{test},b}$ to be the covariates from the b-th test data. The repeated training/testing splitting is as follows:

- Construct B random partitions of the data using training fraction τ and for each b calculate $S_{\text{train},b}(\boldsymbol{X}_{\text{test},b})$.
- The training/testing splitting estimate of statistic S is then $\frac{1}{B}\sum_{b=1}^{B} S_{\text{train},b}(\boldsymbol{X}_{\text{test},b})$.

Foster *et al.* (2011) explored a variety of approaches for estimating quantities similar to subgroup-conditional treatment effects and found bootstrap bias correction approaches to be the least biased and have lower variability than cross-validation based approaches. Foster *et al.* (2011) found that cross-validation approaches tend to underestimate effects of interest. Thus, we advocate the use of the bootstrap bias correction approach, however the training and testing splitting approach is appropriate as well and tends to give more conservative estimates of the subgroup-conditional treatment effects.

3. The personalized package

In this section we provide detailed information about the **personalized** package and how it is utilized in subgroup identification analyses. We begin by providing an outline of the workflow of the **personalized** package. The remainder of this section roughly follows the order of this workflow and explains each function involved in each step. Along the way, key arguments of each of these functions are described in detail with usage examples intermixed. Finally, this section concludes with a demonstration of an entire subgroup identification analysis on a simulated dataset with a multi-category treatment.

3.1. Workflow of subgroup identification analysis

Regardless of the specific modeling choices, the workflow of subgroup identification analyses in the **personalized** package has the following four steps:

1. Construct propensity score function and check propensity score diagnostics (function `check.overlap()`).
2. Fit a subgroup identification model using function `fit.subgroup()`.
3. Estimate the resulting treatment effects among estimated subgroups using function `validate.subgroup()`.
4. Visualize and examine the model (`plot()`), subgroup treatment effects (`print()`), and characteristics of the subgroups (`summarize.subgroups()`).

We will create a simulated dataset where we know the underlying data-generating mechanism. We will use this dataset throughout this paper for illustration. In this simulation, the treatment assignment depends on covariates and hence we must model the propensity score $\pi(\boldsymbol{x}) = \mathsf{P}(T = 1|\boldsymbol{X} = \boldsymbol{x})$. We also assume that larger values of the outcome are better. We generate 1000 samples with 50 covariates and consider continuous, binary, and time-to-event outcomes. The covariates in $\boldsymbol{X}$ in this dataset are generated from a normal distribution and are uncorrelated, the propensity function $\pi(\boldsymbol{x})$ depends only on the 21st and 41st covariates, the optimal treatment rule depends on covariates 3, 11, 1, and 12 including linear terms an an interaction between covariates 1 and 12, and the main effects in the outcome regression model depend on covariates 1, 11, 12, 13, and 15 and include both nonlinear and linear terms.

```
R> library("personalized")
R> set.seed(123)
R> n.obs <- 1000
R> n.vars <- 50
R> x <- matrix(rnorm(n.obs * n.vars, sd = 3), n.obs, n.vars)
```

Here we simulate non-randomized treatment assignment.

```
R> xbetat <- 0.5 + 0.25 * x[, 21] - 0.25 * x[, 41]
R> trt.prob <- plogis(xbetat)
R> trt <- rbinom(n.obs, 1, prob = trt.prob)
```

Here we simulate the differential treatment effect.

```
R> delta <- 0.5 + x[, 2] - 0.5 * x[, 3] - 1 * x[, 11] + 1 * x[, 1] * x[, 12]
```

Now we simulate the main effects and add them to the differential treatment effect and then generate i) continuous, ii) binary, and iii) time-to-event outcomes.

```
R> xbeta <- x[, 1] + x[, 11] - 2 * x[, 12]^2 + x[, 13] + 0.5 * x[, 15]^2
R> xbeta <- xbeta + delta * (2 * trt - 1)
R> trt <- as.factor(ifelse(trt == 1, "Trt", "Ctrl"))
R> y <- xbeta + rnorm(n.obs)
R> y.binary <- 1 * (xbeta + rnorm(n.obs, sd = 2) > 0)
R> surv.time <- exp(-20 - xbeta + rnorm(n.obs, sd = 1))
R> cens.time <- exp(rnorm(n.obs, sd = 3))
R> y.time.to.event <- pmin(surv.time, cens.time)
R> status <- 1 * (surv.time <= cens.time)
```

Note that for the continuous outcomes `y` in the code above, `delta` aligns with (1), however `delta` does not exactly correspond to (1) for the binary outcomes `y.binary` and the time-to-event outcomes `y.time.to.event` above. Still, `delta` in all types of outcomes above drives heterogeneity of treatment effect.

3.2. The `check.overlap()` function

Observational studies

To deal with non-randomized treatment assignment in subgroup identification analysis for observational studies, we usually construct a model for the propensity score, which is the probability of treatment assignment conditional on observed baseline characteristics (Imbens and Rubin 2015). Our package also allows matched analysis for which this step or the `check.overlap()` function is not needed. In the **personalized** package, we need to wrap the propensity score model in a function which inputs covariate values and the treatment statuses and outputs propensity scores between 0 and 1. It is crucial for the **personalized** package to utilize the propensity score model as a function instead of simply a vector of probabilities. Later in this paper when we seek to evaluate the subgroup treatment effects, we must use either bootstrap resampling or repeated training and test splitting of our data. In both of these approaches we need to re-fit our subgroup identification model, including the propensity score model and hence it would be invalid to assume the propensity scores remain constant for each bootstrap iteration. A simple example of how one constructs their propensity score function is as follows.

```
R> propensity.func <- function(x, trt) {
+    data.fr <- data.frame(trt = trt, x)
+    propensity.model <- glm(trt ~ ., family = binomial(), data = data.fr)
+    pi.x <- predict(propensity.model, type = "response")
+    return(pi.x)
+  }
R> propensity.func(x, trt)[101:105]

      101       102       103       104       105
0.2251357 0.2786683 0.9021204 0.4400091 0.8250830

R> trt[101:105]

[1] Ctrl Ctrl Trt  Trt  Trt
Levels: Ctrl Trt
```

The above function uses a binomial generalized linear model (GLM) as the propensity score model and then uses the `predict()` function to return the estimated propensity scores as a vector.

To assess the positivity assumption, propensity scores should be checked to ensure sufficient overlap between treatment groups. This is a requirement for valid use of propensity scores. The **personalized** package offers a visual aid for checking overlap via the `check.overlap()` function, which plots densities or histograms of the propensity scores for each of the treatment groups. The following code generates Figure 1:

```
R> check.overlap(x, trt, propensity.func)
```

To help assess whether there is sufficient overlap of the propensity score distributions between treated and untreated, the user should check whether the regions near 0 or 1 where there is either an area where there is a positive density of propensity scores for the treatment group but not the control group or for the control group and not the treatment group. Figure 1 illustrates the data has reasonable overlap, however there is a slight region near 0 with a positive density of propensity scores for the control group but no density of propensity scores for the treatment group. One may consider corrective measures to mitigate this. In the

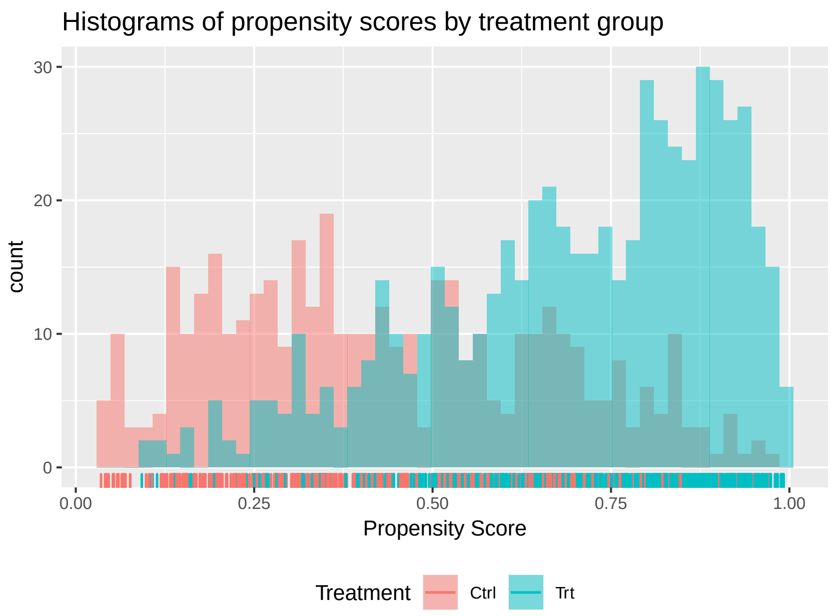

Figure 1: Histograms illustrating overlap of propensity scores.

presence of insufficient overlap, techniques such as those proposed in Crump, Hotz, Imbens, and Mitnik (2009) may be utilized. Further discussion on identification of support overlap issues and approaches for mitigating these issues can be found in Caliendo and Kopeinig (2008) and Garrido, Kelley, Paris, Roza, Meier, Morrison, and Aldridge (2014).

Randomized controlled trials

If the data to be analyzed come from a randomized controlled trial, it is still valid to construct a propensity score model as above, but is not necessary. If the modeler knows that patients were randomized to the treatment group with probability 0.5, for example, the propensity function can simply be constructed as the following:

```
R> constant.propensity.func <- function(x, trt) 0.5
```

3.3. The `fit.subgroup()` function

The `fit.subgroup()` function is the main workhorse of the **personalized** package. It provides fitting capabilities for a wide range of subgroup identification models for different types of outcomes. We will first show a basic usage of the `fit.subgroup()` function and then provide detailed information about its arguments and more involved examples.

A basic usage of `fit.subgroup()` for continuous outcomes based on the A-learning method is as follows. The final argument, `nfolds`, given to `fit.subgroup()` is an argument for the underlying fitting function, `glmnet()`.

```
R> subgrp.model <- fit.subgroup(x = x, y = y, trt = trt,
+    propensity.func = propensity.func, method = "a_learning",
+    loss = "sq_loss_lasso", nfolds = 10)
R> summary(subgrp.model)
```

```
family:    gaussian
loss:      sq_loss_lasso
method:    a_learning
cutpoint:  0
propensity
function:  propensity.func

benefit score: f(x),
Trt recom = Trt*I(f(x)>c)+Ctrl*I(f(x)<=c) where c is 'cutpoint'

Average Outcomes:
                Recommended Ctrl   Recommended Trt
Received Ctrl  -7.8102 (n = 171) -18.589 (n = 239)
Received Trt  -18.9831 (n = 258) -7.5232 (n = 332)

Treatment effects conditional on subgroups:
Est of E[Y|T=Ctrl,Recom=Ctrl]-E[Y|T=/=Ctrl,Recom=Ctrl]
                                     11.1729 (n = 429)
    Est of E[Y|T=Trt,Recom=Trt]-E[Y|T=/=Trt,Recom=Trt]
                                     11.0658 (n = 571)

NOTE: The above average outcomes are biased estimates of
      the expected outcomes conditional on subgroups.
      Use 'validate.subgroup()' to obtain unbiased estimates.

---------------------------------------------------

Benefit score quantiles (f(X) for Trt vs Ctrl):
     0%     25%     50%     75%    100%
-18.144  -2.746   1.020   4.298  16.332

---------------------------------------------------

Summary of individual treatment effects:
E[Y|T=Trt, X] - E[Y|T=Ctrl, X]

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-18.1440  -2.7457   1.0201   0.8175   4.2980  16.3316

---------------------------------------------------

4 out of 50 interactions selected in total by the lasso (cross validation
criterion).

The first estimate is the treatment main effect, which is always selected.
Any other variables selected represent treatment-covariate interactions.
```

```
            Trt     V2      V3    V11    V13
Estimate 0.7957 1.2542 -0.5189 -0.884 0.5292
```

The above code fits a model with linear interaction terms with a squared error loss as $M(\cdot, \cdot)$ with a lasso penalty for variable selection. The squared error loss is used due to the outcome `y` being continuous. The A-learning method is used for demonstrative purposes and in this situation, the weighting method could have been used as well. The `nfolds` argument is passed to the underlying fitting function `cv.glmnet()` from the **glmnet** package. The output provides some basic summary statistics of the resulting subgroups, benefit scores, and estimated conditional treatment effects for each sample and shows the estimated interaction coefficients.

Explanation of major function arguments

`x` The argument `x` is for the design matrix. Each column of `x` corresponds to a variable to be used in the model for $\Delta(\mathbf{X})$ and each row of `x` corresponds to an observation. Every variable in `x` will be used for the subgroup identification model (however some variables may be removed if a variable selection procedure is specified for `loss`).

`y` The argument `y` is for the response vector. Each element in `y` is a patient observation. In the case of time-to-event outcomes `y` should be specified as a '`Surv`' object of the **survival** package (Therneau 2021). For example the user should specify `y = Surv(time, status)`, where `time` is the observed time and `status` is an indicator that the observed time is the survival time.

`trt` The argument `trt` corresponds to the vector of observed treatment statuses. The vector `trt` can either be a character vector specifying the levels of the treatments (e.g., `"Trt"` vs. `"Ctrl"`), a factor vector, or an integer vector (e.g., for binary treatment, 1 or 0 in the i-th position indicates patient i received the treatment or control). For character vectors or integer vectors it is assumed that the first level alphabetically or numerically, respectively, is the reference treatment in the sense that the estimated benefit score will represent the benefit of the second treatment level with respect to the reference level. For example, if `trt` is a character vector with two treatment options `"Trt"` and `"Ctrl"`, the estimated benefit score reflects the benefit of `"Trt"` versus `"Ctrl"` in the sense that positive estimated benefit scores indicate `"Trt"` is preferable to `"Ctrl"`. For a factor vector, the first level of the factor will be the reference treatment. Without specifying otherwise, for `trt` vectors with more than 2 treatment levels, the reference treatment will be chosen in the same way. However, the user may specify which level of the treatment should be the reference via the `reference.trt` argument. For example, if `trt` has the levels `c("TrtA", "TrtB", "Ctrl")`, setting `reference.trt = "Ctrl"` will ensure that `"Ctrl"` is the reference level.

`propensity.func` The argument `propensity.func` corresponds to a function which returns a propensity score. While it seems cumbersome to have to specify a function instead of a vector of probabilities, it is crucial for later validation for the propensity scores to be re-estimated using the resampled or sampled data (this will be explained further in the section below for the `validate.subgroup()` function). The user should specify a function which inputs two arguments: `trt` and `x`, where `trt` corresponds to the `trt` argument for the `fit.subgroup()` function and `x` corresponds to the `x` argument for the `fit.subgroup()` function. The function supplied to the `propensity.func` argument should contain code that uses `x` and `trt` to fit a propensity score model and then return an estimated propensity score for each observation in `x`. If there are many covariates, the modeler may wish to use variable selection techniques in constructing the propensity score model. In the following code we construct the wrapper function for the propensity score model, which is a logistic regression model with the lasso penalty where the tuning parameter is selected by 10-fold cross-validation using the `cv.glmnet()` function of the **glmnet** package (Friedman *et al.* 2021):

```
R> propensity.func.lasso <- function(x, trt) {
+    propens.model <- cv.glmnet(y = trt, x = x, family = "binomial")
+    pi.x <- predict(propens.model, s = "lambda.min", newx = x,
+      type = "response")[, 1]
+    pi.x
+  }
```

For randomized controlled trials with equal probability of assignment to treatment and control, the user can simply define `propensity.func` as:

```
R> propensity.func.const <- function(x, trt) 0.5
```

which always returns the constant 1/2.

For cases with multi-category treatments, the user must specify a propensity function that returns $\mathsf{P}(T = T_i|\mathbf{X} = \mathbf{x})$ for patient i. In other words, it should return the probability of receiving the treatment that was actually received for each patient. For example, the below code uses `nnet::multinom`, whose `predict` method returns a matrix of probabilities with one column for each treatment level. We produce code below that returns the probability corresponding to the treatment that was actually observed for each observation.

```
R> propensity.func.multinom <- function(x, trt) {
+    require("nnet")
+    df <- data.frame(trt = trt, x)
+    mfit <- nnet::multinom(trt ~ . -trt, data = df)
+    propens <- predict(mfit, type = "probs")
+    if (is.factor(trt)) {
+      values <- levels(trt)[trt]
+    } else {
+      values <- trt
+    }
+    probs <- propens[cbind(1:nrow(propens),
+      match(values, colnames(propens)))]
+    probs
+  }
```

Optionally the user can specify the function to return a matrix of treatment probabilities, however, the columns *must* be ordered by the levels of `trt`. An example of this is the following, where we ensure that the columns are ordered correctly:

`loss` prefix	Outcomes	$M(y, v)$
`"sq_loss"`	C/B/CT	$(y - v)^2$
`"logistic_loss"`	B	$-[yv - \log(1 + \exp\{-v\})]$
`"owl_logistic"`†	C/B/CT	$y \log(1 + \exp\{-v\})$
`"owl_hinge"`†	C/B/CT	$y \max(0, 1 - v)$
`"owl_logistic_flip"`	C/B/CT	$\lvert y\rvert \log(1 + \exp\{-\mathrm{sign}(y)v\})$
`"owl_hinge_flip"`	C/B/CT	$\lvert y\rvert \max(0, 1 - \mathrm{sign}(y)v)$
`"poisson_loss"`†	CT	$-[yv - \exp(v)]$
`"cox_loss"`	TTE	$-\left\{\int_0^\tau (v - \log[\mathsf{E}\{e^v I(X \geq u)\}])\,\mathrm{d}N(u)\right\}$ where $y = (X, \delta) = \{\widetilde{X} \wedge C, I(\widetilde{X} \leq t)\}$, $\widetilde{X}$ is the survival time, C is the censoring time, $N(t) = I(\widetilde{X} \leq t)\delta$, and τ is a fixed point such that $\mathsf{P}(X \geq \tau) > 0$.

†The outcomes need to be non-negative.

Table 2: Listed above are the forms of the loss function $M(y, v)$ available in the **personalized** package. In the outcomes column, "C" indicates a loss is available for continuous outcomes, "B" for binary outcomes, "CT" for count outcomes, and "TTE" for time-to-event outcomes. Note that positive continuous outcomes may be used for `"poisson_loss"` as well. In general there are fewer restrictions in theory about the types of outcomes used for the above losses, however the imposed restrictions in this package are due to implementation limitations.

```
R> propensity.func.multinom <- function(x, trt) {
+    require("nnet")
+    df <- data.frame(trt = trt, x)
+    mfit <- multinom(trt ~ . -trt, data = df)
+    propens <- predict(mfit, type = "probs")
+    if (is.factor(trt)) {
+      levels <- levels(trt)
+    } else {
+      levels <- sort(unique(trt))
+    }
+    probs <- propens[, match(levels, colnames(propens))]
+    probs
+  }
```

For more information on the construction of propensity scores for multi-category treatments, see McCaffrey, Griffin, Almirall, Slaughter, Ramchand, and Burgette (2013).

`loss` The `loss` argument specifies the combination of M function (i.e., loss function) and underlying model $f(\boldsymbol{X})$. The name of each possible value for `loss` has two parts: The first part corresponds to the M function and the second part corresponds to $f(\boldsymbol{X})$ and whether variable selection via the lasso is used. The available M functions are listed in Table 2.

All `loss` options that have `"lasso"` in their suffix use $f(\boldsymbol{X}) = \boldsymbol{X}^\top\boldsymbol{\beta}$ and have the penalty term $\sum_{j=1}^p |\beta_j|$ added to the overall objective function $L_W(f)$ or $L_A(f)$. Adding the penalty term makes the benefit score estimate $\hat{f}(\boldsymbol{X}) = \boldsymbol{X}^\top \hat{\boldsymbol{\beta}}$ sparse in the sense that some elements of $\hat{\boldsymbol{\beta}}$ will be exactly zero, allowing a simpler form of the benefit score. An example is `"sq_loss_lasso"`, which corresponds to using $M(y, v) = (y - v)^2$, a linear form of f, i.e., $f(\boldsymbol{X}) = \boldsymbol{X}^\top \beta$, and an additional penalty term $\sum_{j=1}^p |\beta_j|$ added to the loss function for variable selection. All options containing `"lasso"` in the name use the `cv.glmnet()` function of the **glmnet** package (Friedman *et al.* 2021) for the underlying model fitting and variable selection. A K-fold cross-validation is used to select the penalty tuning parameter. Please see the documentation of `cv.glmnet()` for information about other arguments which can be passed to it.

Any options for `loss` which end with `"lasso_gam"` have a two-stage model. Variables are selected using a linear or GLM in the first stage and then the selected variables are used in a generalized additive model in the second stage. Univariate nonparametric smoother terms are used in the second stage for all continuous variables. Binary variables are used as linear terms in the model. All `loss` options containing `"gam"` in the name use the `gam()` function of the R package **mgcv** (Wood 2017, 2019). Please see the documentation of `gam()` for information about other arguments which can be passed to it.

All options that end in `"gbm"` use gradient-boosted decision trees models for $f(\boldsymbol{X})$. Such machine learning models can provide more flexible forms of estimation by essentially using a sum of decision trees models. However, these "black box" models can be more challenging to interpret. The `gbm`-based models are fit using the **gbm** R package (Greenwell, Boehmke, Cunningham, and GBM Developers 2020). Please see the documentation for the `gbm()` function of the **gbm** package for more details on the possible arguments. Tuning the values of the hyperparameters `shrinkage`, `n.trees`, and `interaction.depth` is crucial for a successful gradient-boosting model. These arguments can be passed to the `fit.subgroup()` function. By default, when `gbm`-based models are used, a plot of the cross-validation error versus the number of trees is displayed. If this plot has values which are still decreasing at the maximum value of the number of trees, then it is recommended to either increase the number of trees (`n.trees`), the maximum tree depth (`interaction.depth`), or the step size of the algorithm (`shrinkage`).

The loss `"owl_hinge"` options are based on the hinge loss function and thus correspond to support vector machine type of optimization procedures. Optimization of hinge-based losses is done via the **kernlab** package (Karatzoglou, Smola, Hornik, and Zeileis 2004; Karatzoglou, Smola, and Hornik 2019). As such, the underlying model for $f(\boldsymbol{X})$ depends on the kernel chosen by the user. A linear kernel will yield a linear decision rule $f(\boldsymbol{X})$, whereas a nonlinear kernel such as the Gaussian radial basis function kernel will yield a more flexible, nonlinear decision rule. Available kernels are listed in the **kernlab** package and can be displayed by `?kernels`.

The outcome-weighted learning based losses with `"flip"` in their name allow for non-positive outcomes. In these cases they may offer substantial finite sample efficiency gains compared with using the original outcome-weighted learning losses and shifting the response such that it is positive.

`method` The `method` argument is used to specify whether the weighting or A-learning method is used. Specify `"weighting"` for the weighting method that uses $L_W(f)$ and `"a_learning"` for the A-learning method that uses $L_A(f)$.

`match.id` This argument allows the user to specify that the analysis dataset is based on matched groups of cases and controls. If used, it should be either a character, factor, or integer vector with length equal to the number of observations in `x` indicating which patients are in which matched groups. Defaults to `NULL` and assumes the samples are not from a matched cohort. Matched case-control groups can be created using any method such as propensity score matching, optimal matching, etc. (Imbens and Rubin 2015). If each case is matched with a control or multiple controls, this would indicate which case-control pairs or groups go together. If `match.id` is supplied, then it is unnecessary to specify a function via the `propensity.func` argument. A quick usage example is: If the first patient is a case and the second and third are controls matched to it, and the fourth patient is a case and the fifth through seventh patients are matched with it, then the user should specify `match.id = c(1, 1, 1, 2, 2, 2, 2)` or `match.id = rep(c("Grp1", "Grp2"), c(3, 4))`.

`augment.func` The `augment.func` argument is used to allow the user to specify an efficiency augmentation function. The basic idea of efficiency augmentation is to construct a model for the main effects of the outcome model and shift the outcome based on these main effects. The resulting estimator based on the shifted outcome can be more efficient than using the outcome itself.

For the same reason that the `propensity.func` must be specified as a function, the user should specify a wrapper function for `augment.func` which inputs the covariate information `x` and the outcome `y` and outputs a prediction for the outcome for each observation in `x`. The predictions should be returned on the link scale, in other words on the scale of the linear predictors. The augmentation function may be from a nonlinear or nonparametric model, however the predictions should still be returned on the link scale. An example of an augmentation function that uses linear regression with a lasso penalty for this model is as follows:

```
R> augment.func.simple <- function(x, y) {
+    cvmod <- cv.glmnet(y = y, x = x, nfolds = 10)
+    predictions <- predict(cvmod, newx = x, s = "lambda.min")
+    predictions
+  }
```

A more involved example that models the full conditional outcome $\mathsf{E}[Y|T, \mathbf{X}]$ and integrates over the treatment levels is given by the following code, which obtains predictions for each observation under both treatment levels and averages these predictions:

```
R> augment.func <- function(x, y, trt) {
+    data <- data.frame(x, y, trt = ifelse(trt == "Trt", 1, -1))
+    xm <- model.matrix(y ~ trt * x - 1, data = data)
+    cvmod <- cv.glmnet(y = y, x = xm)
+    data$trt <- 1
+    xm1 <- model.matrix(y ~ trt * x - 1, data = data)
+    preds_1  <- predict(cvmod, xm1, s = "lambda.min")
+    data$trt <- -1
+    xm2 <- model.matrix(y ~ trt * x - 1, data = data)
+    preds_n1  <- predict(cvmod, xm2, s = "lambda.min")
+    return(0.5 * (preds_1 + preds_n1))
+  }
```

For binary outcomes, one must define the augmentation function such that it returns predictions on the link scale as follows:

```
R> augment.func.bin <- function(x, y) {
+    cvmod <- cv.glmnet(y = y, x = x, family = "binomial")
+    predict(cvmod, newx = x, s = "lambda.min", type = "link")
+  }
```

Then the defined augmentation function can be used in `fit.subgroup()` by passing the function to the argument `augment.func`. A usage example using the above-defined augmentation function is the following:

```
R> subgrp.model.eff <- fit.subgroup(x = x, y = y, trt = trt,
+    propensity.func = propensity.func, loss = "sq_loss_lasso",
+    augment.func = augment.func, nfolds = 10)
R> summary(subgrp.model.eff)

family:    gaussian
loss:      sq_loss_lasso
method:    weighting
cutpoint:  0
augmentation
function: augment.func
propensity
function:  propensity.func

benefit score: f(x),
Trt recom = Trt*I(f(x)>c)+Ctrl*I(f(x)<=c) where c is 'cutpoint'

Average Outcomes:
                Recommended Ctrl    Recommended Trt
Received Ctrl  -8.6343 (n = 178) -18.0961 (n = 232)
Received Trt  -19.7439 (n = 254)  -7.0398 (n = 336)

Treatment effects conditional on subgroups:
Est of E[Y|T=Ctrl,Recom=Ctrl]-E[Y|T=/=Ctrl,Recom=Ctrl]
                                     11.1095 (n = 432)
    Est of E[Y|T=Trt,Recom=Trt]-E[Y|T=/=Trt,Recom=Trt]
                                     11.0563 (n = 568)

NOTE: The above average outcomes are biased estimates of
      the expected outcomes conditional on subgroups.
      Use 'validate.subgroup()' to obtain unbiased estimates.
```

```
---------------------------------------------------

Benefit score quantiles (f(X) for Trt vs Ctrl):
     0%      25%      50%      75%     100%
-12.750  -1.885   0.832   3.248  10.467

---------------------------------------------------

Summary of individual treatment effects:
E[Y|T=Trt, X] - E[Y|T=Ctrl, X]

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-25.500  -3.770   1.664   1.363   6.496  20.934

---------------------------------------------------

13 out of 50 interactions selected in total by the lasso (cross validation
criterion).

The first estimate is the treatment main effect, which is always selected.
Any other variables selected represent treatment-covariate interactions.

            Trt     V1     V2      V3     V8      V9     V11    V13    V17
Estimate 0.7119 0.2294 0.6364 -0.3792 -0.016 -0.0467 -0.7971 0.4577 0.0512
             V27    V36     V42     V43    V50
Estimate -0.1035 0.0489 -0.1526 -0.1059 0.0434
```

From the online supplementary material of Chen *et al.* (2017), the optimal efficiency augmentation function may depend on the treatment statuses. Hence the user is allowed to specify `augment.func` as additionally a function of `trt`, i.e., `augment.func <- function(x, y, trt)`.

`fit.custom.loss` The `fit.custom.loss` argument allows the user to provide a function which *minimizes* a custom loss function for use in the `fit.subgroup()` function. The loss function, $\mathbf{M}(y, v)$, to be minimized must meet the criteria outlined in Section 2.2. The user must provide as `fit.custom.loss` a function which minimizes a sample weighted version of the loss function and returns a list with the solution of the minimization in addition to a function which takes covariates as an argument and returns predictions of the benefit score $\hat{f}(\mathbf{x})$ under the estimator resulting from the minimization of the custom loss.

If provided, this function should take the modified design matrix and the responses as argument s and optimize a custom weighted loss function. The provided function *must* be a function with the following arguments:

- `x` – design matrix.
- `y` – vector of responses.
- `weights` – vector for observations weights. The underlying loss function *must* have samples weighted according to this vector. See the example below.

- `...` – additional arguments passed via `...`. This can be used so that users can specify more arguments to the underlying fitting function via `fit.subgroup()` if so desired.

The provided function *must* return a list with the following elements:

- `predict` – a function that inputs a design matrix and a `type` argument for the type of predictions and outputs a vector of predictions on the scale of the linear predictor. Note that the matrix provided to `fit.custom.loss` has a column appended to the first column of `x` corresponding to the treatment main effect. Thus, the prediction function should deal with this, e.g., `predict(model, cbind(1, x))`.
- `model` – a fitted model object returned by the underlying fitting function. This can be an arbitrary R object.
- `coefficients` – if the underlying fitting function yields a vector of coefficient estimates, they should be provided here.

The provided function can also optionally take the following arguments which may be optionally used in the custom fitting routine:

- `match.id` – vector of case/control cluster identifiers. This is useful if cross-validation is used in the underlying fitting function in which case it is advisable to sample whole clusters randomly instead of individual observations.
- `offset` – if efficiency augmentation is used, the predictions from the outcome model from `augment.func` will be provided via the `offset` argument, which can be used as an offset in the underlying fitting function as a means of incorporating the efficiency augmentation model's predictions.
- `trt` – vector of treatment statuses.
- `family` – family of outcome.

An example of `fit.custom.loss` is a minimization of the exponential loss $\mathbf{M}(y, v) = y\exp(-v)$ for positive outcomes. In the following code, we first define the weighted loss function with a linear form for the benefit score as the function `expo.loss`. We then use the `optim()` function to minimize this function to obtain coefficient estimates and then define `pred` to be a function which returns predicted benefit scores. The function finally returns a list with the prediction function, the returned optimization object, and the estimated coefficients.

```
R> fit.expo.loss <- function(x, y, weights, ...) {
+    expo.loss <- function(beta, x, y, weights) {
+      sum(weights * y * exp(-drop(x %*% beta)))
+    }
+    opt <- optim(rep(0, NCOL(x)), fn = expo.loss, x = x, y = y,
+       weights = weights)
+    coefs <- opt$par
+    pred <- function(x, type = "response") {
+      cbind(1, x) %*% coefs
+    }
+    list(predict = pred, model = opt, coefficients = coefs)
+  }
```

`larger.outcome.better` The argument `larger.outcome.better` is a Boolean variable indicating whether larger values of the outcome are better or preferred. If the argument `larger.outcome.better = TRUE`, then `fit.subgroup()` will seek to estimate subgroups in a way that maximizes the population average outcome and if `larger.outcome.better = FALSE`, `fit.subgroup()` will seek to minimize the population average outcome.

`reference.trt` As already mentioned for `trt`, the user may specify which level of the treatment should be the reference via the `reference.trt` argument. For example, if `trt` has the levels `c("TrtA", "TrtB", "Ctrl")`, setting `reference.trt = "Ctrl"` will ensure that `"Ctrl"` is the reference level. This argument is not used for multi-category treatment fitting with OWL-type losses, as the underlying multinomial outcome-weighted model is parameterized such that there is not a reference treatment group. This parameterization is described in Friedman, Hastie, and Tibshirani (2010).

`cutpoint` The cutpoint is the numeric value of the benefit score $f(\boldsymbol{X})$ above which patients will be recommended the treatment. In other words for outcomes where larger values are better and a cutpoint with value c if $f(\boldsymbol{x}) > c$ for a patient with covariate values $\boldsymbol{X} = \boldsymbol{x}$, then they will be recommended to have the treatment instead of recommended the control. If lower values are better for the outcome, c will be the value below which patients will be recommended the treatment (i.e., a patient will be recommended the treatment if $f(\boldsymbol{x}) < c$). By default the cutpoint value is 0. Users may wish to increase this value if there are limited resources for treatment allocation. The cutpoint argument is available for multi-category treatments and is still a single value applied to each comparison with the reference treatment.

The user can also set `cutpoint = "median"`, which will use the median value of the benefit scores as the cutpoint. Similarly, the user can set specific quantile values via `"quantx"` where `"x"` is a number between 0 and 100 representing the quantile value; e.g., `cutpoint = "quant75"` will use the 75th percent upper quantile of the benefit scores as the cutpoint value.

`retcall` The argument `retcall` is a Boolean variable which indicates whether to return the arguments passed to `fit.subgroup()`. It must be set to `TRUE` if the user wishes to later validate the fitted model object from `fit.subgroup()` using the `validate.subgroup()` function. This is necessary because when `retcall = TRUE`, the design matrix `x`, response `y`, and treatment vector `trt` must be re-sampled in either the bootstrap procedure or training and testing resampling procedure of `validate.subgroup()`. The only time when `retcall` should be set to `FALSE` is when the design matrix is too big to be stored in the fitted model object.

`...` The argument `...` is used to pass arguments to the underlying modeling functions. For example, if the lasso is specified in the `loss` argument, `...` is used to pass arguments to the `cv.glmnet()` function from the **glmnet** package. If `gam` is present in the name for the `loss` argument, the underlying model is fit using the `gam()` function of **mgcv**, so arguments to `gam()` can be passed using `...`. The only tricky part for `gam()` is that it also has an argument titled `method` and hence instead, to change the `method` argument of `gam()`, the user can pass values using `method.gam` which will then be passed as the argument for `method` in the `gam()`

function. For all `loss` options with `"hinge"`, this will be passed to both `weighted.ksvm()` from the **personalized** package and `ipop` from the **kernlab** package.

Continuous outcomes

The `loss` argument options that are available for continuous outcomes are:

- `"sq_loss_lasso"`
- `"owl_logistic_loss_lasso"`
- `"owl_hinge_loss"`
- `"owl_logistic_flip_loss_lasso"`
- `"owl_hinge_flip_loss"`
- `"sq_loss_gam"`
- `"owl_logistic_loss_gam"`
- `"owl_logistic_flip_loss_gam"`
- `"sq_loss_lasso_gam"`
- `"owl_logistic_loss_lasso_gam"`
- `"owl_logistic_flip_loss_lasso_gam"`
- `"sq_loss_gbm"`

Note that the loss options `"owl_logistic_loss_lasso"`, `"owl_logistic_loss_gam"`, and `"owl_logistic_loss_lasso_gam"` require the outcome to be positive whereas the corresponding options with `"_flip_"` in them have no such requirement. Similarly, the option `"owl_hinge_loss"` requires the outcome to be positive whereas `"owl_hinge_flip_loss"` does not.

Flexible gradient-boosted decision trees models can also be used. A typical usage of such models for continuous outcomes is as follows, where the last 4 arguments provided to function `fit.subgroup()` are arguments to be passed to the underlying fitting function, `gbm()`.

```
R> subgrp.model.gbm <- fit.subgroup(x = x, y = y, trt = trt,
+    propensity.func = propensity.func.lasso, loss = "sq_loss_gbm",
+    shrinkage = 0.025, n.trees = 1000, interaction.depth = 2, cv.folds = 5)
```

Binary outcomes

All loss options for continuous outcomes can also be used for binary outcomes. Additionally, the `loss` argument options that are exclusively available for binary outcomes are:

- `"logistic_loss_lasso"`
- `"logistic_loss_lasso_gam"`

- `"logistic_loss_gam"`
- `"logistic_loss_gbm"`

```
R> subgrp.bin <- fit.subgroup(x = x, y = y.binary, trt = trt,
+    propensity.func = propensity.func.lasso,
+    loss = "logistic_loss_lasso", nfolds = 10)
```

When gradient-boosted decision trees are used for $f(\boldsymbol{X})$ by the package **gbm**, care must be taken to choose the hyperparameters effectively. Specifically, `shrinkage` (similar to the step-size in gradient descent), `n.trees` (the number of trees to fit), and `interaction.depth` (the maximum depth of each tree) should be tuned according to the data at hand. By default for gradient-boosting models, `fit.subgroup()` plots the cross-validation error versus the number of trees to enable the users to assess their choice of tuning parameters.

Count outcomes

All loss options for continuous outcomes can also be used for count outcomes. Additionally, the `loss` argument options that are exclusively available for count outcomes are:

- `"poisson_loss_lasso"`
- `"poisson_loss_lasso_gam"`
- `"poisson_loss_gam"`
- `"poisson_loss_gbm"`

Time-to-event outcomes

The `loss` argument options that are available for time-to-event outcomes are:

- `"cox_loss_lasso"`
- `"cox_loss_gbm"`

For subgroup identification models for time-to-event outcomes, the user should provide function `fit.subgroup()` with a '`Surv`' object of the **survival** package for `y`. This can be done as follows:

```
R> library("survival")
R> set.seed(123)
R> subgrp.cox <- fit.subgroup(x = x, y = Surv(y.time.to.event, status),
+    trt = trt, propensity.func = propensity.func.lasso,
+    loss = "cox_loss_lasso", nfolds = 10)
```

The subgroup treatment effects are estimated using the restricted mean survival time statistic (Irwin 1949; Zhao and Tsiatis 1997, 1999; Chen and Tsiatis 2001) and can be displayed with the `summary` or `print` methods for the '`subgroup_fitted`' objects returned as follows:

```
R> summary(subgrp.cox)

family:    cox
loss:      cox_loss_lasso
method:    weighting
cutpoint:  0
propensity
function:  propensity.func

benefit score: f(x),
Trt recom = Trt*I(f(x)>c)+Ctrl*I(f(x)<=c) where c is 'cutpoint'

Average Outcomes:
                Recommended Ctrl    Recommended Trt
Received Ctrl 275.7499 (n = 255)  11.9909 (n = 155)
Received Trt  367.1772 (n = 369) 162.6475 (n = 221)

Treatment effects conditional on subgroups:
Est of E[Y|T=Ctrl,Recom=Ctrl]-E[Y|T=/=Ctrl,Recom=Ctrl]
                                    -91.4273 (n = 624)
    Est of E[Y|T=Trt,Recom=Trt]-E[Y|T=/=Trt,Recom=Trt]
                                    150.6566 (n = 376)

NOTE: The above average outcomes are biased estimates of
      the expected outcomes conditional on subgroups.
      Use 'validate.subgroup()' to obtain unbiased estimates.

---------------------------------------------------

Benefit score quantiles (f(X) for Trt vs Ctrl):
      0%      25%      50%      75%     100%
-0.51188 -0.16824 -0.05754  0.07037  0.66802

---------------------------------------------------

Summary of individual treatment effects:
E[Y|T=Trt, X] / E[Y|T=Ctrl, X]
Note: for survival outcomes, the above ratio is
E[g(Y)|T=Trt, X] / E[g(Y)|T=Ctrl, X],
where g() is a monotone increasing function of Y,
the survival time

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.5127  0.9320  1.0592  1.0686  1.1832  1.6684

---------------------------------------------------
```

```
8 out of 49 interactions selected in total by the lasso (cross validation
criterion).

The first estimate is the treatment main effect, which is always selected.
Any other variables selected represent treatment-covariate interactions.

            Trt     V1     V2      V3     V11     V12    V13   V17     V47
Estimate 0.0461 0.0065 0.0473 -0.0101 -0.0207 -0.0014 0.0065 2e-04 -0.0024
            V50
Estimate 0.0176
```

3.4. The `summarize.subgroups()` function for summarizing subgroups

The `summarize.subgroups()` function provides a quick way of comparing the covariate values between the subgroups recommended the treatment and the control respectively. p values for the differences of covariate values between subgroups are computed and adjusted for multiple comparisons using the approach of Hommel (1988). For continuous variables the p values come from t tests and for discrete variables the p values come from chi-squared tests. The p values are computed and used as a means to filter out covariates without meaningful differences between subgroups, however they are not displayed as they do not represent valid statistical inferences due to their post-hoc nature.

```
R> comp <- summarize.subgroups(subgrp.cox)
```

The user can optionally print only the covariates which have "significant" differences between subgroups with a multiple comparisons-adjusted p value below a given threshold like the following:

```
R> print(comp, p.value = 0.01)

    Avg (recom Ctrl) Avg (recom Trt) Ctrl - Trt SE (recom Ctrl)
V1            0.4021         -0.5387     0.9408         0.11842
V2            1.7412         -2.5508     4.2920         0.09051
V11          -0.5269          1.0827    -1.6095         0.11687
V50           0.8054         -1.0275     1.8329         0.12070
    SE (recom Trt)
V1          0.1503
V2          0.1085
V11         0.1500
V50         0.1564
```

3.5. The `validate.subgroup()` function for evaluating identified subgroups

It is crucial to evaluate the findings by assessing the improvement in outcomes with the estimated subgroups. Ideally, the treatment should have a positive impact on the outcome within the subgroup of patients who are recommended the treatment and the control should have a positive impact on the outcome within the subgroup of patients who were not recommended the treatment.

In general it is quite challenging to obtain valid estimates of these effects because usually only one dataset is available. Using data twice, or taking the average outcomes by treatment status within each subgroup (using the same data) to estimate the treatment effects, will yield biased and typically overly-optimistic estimates of the subgroup-specific treatment effects. Therefore, as described in Section 2.6, we use resampling-based procedures to alleviate this phenomenon and hope to estimate these effects reliably. The **personalized** package offers two methods for subgroup treatment effect estimation. Both methods are available via the `validate.subgroup()` function.

Repeated training/test splitting

The first method of subgroup-specific treatment effects available in `validate.subgroup()` is prediction-based and requires multiple replications of data partitioning. For each replication in this procedure, data are randomly partitioned into training and testing portions. Then the subgroup identification model is estimated using the training portion and the subgroup treatment effects are estimated via empirical averages within subgroups using the testing portion. This method requires two arguments to be passed to `validate.subgroup()`. The first argument is `B`, the number of replications and the second argument is `train.fraction`, the proportion of samples used for training. Hence `1 - train.fraction` is the portion of samples used for testing.

The main object which needs to be passed to `validate.subgroup()` is a fitted object returned by `fit.subgroup()`. Note that in order to validate a fitted object from `fit.subgroup()`, the model must be fit with the `fit.subgroup()` argument `retcall` set to `TRUE` because the data passed to `fit.subgroup()` must be accessed. The `validate.subgroup()` function uses the same arguments that were passed to the original call of `fit.subgroup()` for fitting during each replication.

The validation process is carried out by fixing the cutpoint value at the user specified cutpoint from the call to `fit.subgroup()`. However, especially in scenarios with very costly treatments, it may be of interest to investigate the treatment effects within subgroups defined by different cutpoints along the range of the benefit score. To simultaneously run the validation procedure for subgroups defined by different cutpoints of the benefit score, the user can specify a vector of benefit score quantiles to `validate.subgroup()` via the `benefit.score.quantiles` argument. For example, setting `benefit.score.quantiles = c(0.5, 0.75)` will yield validation results for subgroups defined by a median cutoff value for the benefit score and a cutoff value at the 75th quantile of the benefit score, the latter of which will result in a smaller subgroup assigned to the treatment that will ideally have a larger treatment effect. The default value for `benefit.score.quantiles` is the vector `c(1/6, 2/6, 3/6, 4/6, 5/6)`. The results of this can be accessed by setting the argument `type = "conditional"` for the `plot` method of '`subgroup_validated`' objects or by specifying a vector of indexes via the argument `which.quant` of the `print` method of '`subgroup_validated`' objects.

```
R> class(subgrp.model.eff)

[1] "subgroup_fitted"
```

Here the argument `B` specifies the number of replications, `method` indicates what estimation method to use, and `benefit.score.quantiles` specifies which quantiles of the benefit score to use as cutpoints in addition to 0.

```
R> validation.eff <- validate.subgroup(subgrp.model.eff, B = 25,
+    method = "training_test_replication",
+    benefit.score.quantiles = c(0.5, 0.75, 0.9), train.fraction = 0.75)
R> validation.eff

family:  gaussian
loss:    sq_loss_lasso
method:  weighting

validation method:  training_test_replication
cutpoint:           0
replications:       25

benefit score: f(x),
Trt recom = Trt*I(f(x)>c)+Ctrl*I(f(x)<=c) where c is 'cutpoint'

Average Test Set Outcomes:
                               Recommended Ctrl
Received Ctrl -11.2162 (SE = 4.9659, n = 44.64)
Received Trt   -16.4652 (SE = 2.8617, n = 64.4)
                                Recommended Trt
Received Ctrl -16.0059 (SE = 3.1811, n = 58.08)
Received Trt   -9.3996 (SE = 2.0931, n = 82.88)

Treatment effects conditional on subgroups:
Est of E[Y|T=Ctrl,Recom=Ctrl]-E[Y|T=/=Ctrl,Recom=Ctrl]
                       5.249 (SE = 6.5083, n = 109.04)
    Est of E[Y|T=Trt,Recom=Trt]-E[Y|T=/=Trt,Recom=Trt]
                      6.6063 (SE = 4.0641, n = 140.96)

Est of
E[Y|Trt received = Trt recom] - E[Y|Trt received =/= Trt recom]:
5.4049 (SE = 2.7976)
```

Note that when a larger quantile cutoff is used fewer patients are recommended the treatment, however the treatment effect among those recommended the treatment is much larger.

```
R> print(validation.eff, which.quant = c(2, 3))

family:  gaussian
loss:    sq_loss_lasso
method:  weighting
```

```
validation method:  training_test_replication
cutpoint:           Quant_75
replications:       25

benefit score: f(x),
Trt recom = Trt*I(f(x)>c)+Ctrl*I(f(x)<=c) where c is 'cutpoint'

Average Test Set Outcomes:
                                Recommended Ctrl
Received Ctrl   -13.6323 (SE = 3.057, n = 76.52)
Received Trt  -14.2462 (SE = 1.4602, n = 110.48)
                               Recommended Trt
Received Ctrl -15.5259 (SE = 4.2005, n = 26.2)
Received Trt   -7.2177 (SE = 4.0275, n = 36.8)

Treatment effects conditional on subgroups:
Est of E[Y|T=Ctrl,Recom=Ctrl]-E[Y|T=/=Ctrl,Recom=Ctrl]
                        0.6139 (SE = 3.3128, n = 187)
    Est of E[Y|T=Trt,Recom=Trt]-E[Y|T=/=Trt,Recom=Trt]
                          8.3082 (SE = 6.048, n = 63)

Est of E[Y|Trt received = Trt recom] - E[Y|Trt received =/= Trt recom]:
2.5018 (SE = 2.484)

<=================================================>

family:  gaussian
loss:    sq_loss_lasso
method:  weighting

validation method:  training_test_replication
cutpoint:           Quant_90
replications:       25

benefit score: f(x),
Trt recom = Trt*I(f(x)>c)+Ctrl*I(f(x)<=c) where c is 'cutpoint'

Average Test Set Outcomes:
                                Recommended Ctrl
Received Ctrl  -13.8475 (SE = 2.8093, n = 91.88)
Received Trt  -13.0648 (SE = 1.3536, n = 133.12)
                                Recommended Trt
Received Ctrl -16.2354 (SE = 6.8779, n = 10.84)
Received Trt   -7.3182 (SE = 8.6173, n = 14.16)

Treatment effects conditional on subgroups:
Est of E[Y|T=Ctrl,Recom=Ctrl]-E[Y|T=/=Ctrl,Recom=Ctrl]
```

```
                        -0.7826 (SE = 3.4515, n = 225)
    Est of E[Y|T=Trt,Recom=Trt]-E[Y|T=/=Trt,Recom=Trt]
                         8.9172 (SE = 11.4587, n = 25)

Est of E[Y|Trt received = Trt recom] - E[Y|Trt received =/= Trt recom]:
0.1748 (SE = 3.3348)
```

Bootstrap bias correction

The second method for estimation of subgroup-conditional treatment effects described in Section 2.6 and available in `validate.subgroup()` is the bootstrap bias correction method. The bootstrap bias correction method can be accessed via the `validate.subgroup()` function as follows:

```
R> validation.boot <- validate.subgroup(subgrp.model.eff, B = 100,
+    method = "boot_bias_correction")
R> validation.boot

family:  gaussian
loss:    sq_loss_lasso
method:  weighting

validation method:  boot_bias_correction
cutpoint:           0
replications:       100

benefit score: f(x),
Trt recom = Trt*I(f(x)>c)+Ctrl*I(f(x)<=c) where c is 'cutpoint'

Average Bootstrap Bias-Corrected Outcomes:
                                Recommended Ctrl
Received Ctrl  -11.4446 (SE = 1.875, n = 177.19)
Received Trt  -17.7519 (SE = 1.9165, n = 253.83)
                                 Recommended Trt
Received Ctrl -15.6872 (SE = 1.9325, n = 231.04)
Received Trt   -8.7282 (SE = 1.1628, n = 337.94)

Treatment effects conditional on subgroups:
Est of E[Y|T=Ctrl,Recom=Ctrl]-E[Y|T=/=Ctrl,Recom=Ctrl]
                      6.3073 (SE = 2.3743, n = 431.02)
    Est of E[Y|T=Trt,Recom=Trt]-E[Y|T=/=Trt,Recom=Trt]
                       6.959 (SE = 2.1228, n = 568.98)

Est of
E[Y|Trt received = Trt recom] - E[Y|Trt received =/= Trt recom]:
6.5025 (SE = 1.7104)
```

Evaluating performance of subgroup-specific treatment effect estimation

We now generate an independent dataset from the same data-generating mechanism of the simulation in order to evaluate how well the subgroup-specific treatment effects are estimated by `validate.subgroup()`.

```
R> x.test <- matrix(rnorm(10 * n.obs * n.vars, sd = 3), 10 * n.obs, n.vars)
R> xbetat.test <- 0.5 + 0.25 * x.test[, 21] - 0.25 * x.test[, 41]
R> trt.prob.test <- plogis(xbetat.test)
R> trt.test <- rbinom(10 * n.obs, 1, prob = trt.prob.test)
R> delta.test <- 0.5 + x.test[, 2] - 0.5 * x.test[,3] -  x.test[, 11] +
+    x.test[, 1] * x.test[, 12]
R> xbeta.test <- x.test[, 1] + x.test[, 11] - 2 * x.test[, 12]^2 +
+    x.test[, 13] + 0.5 * x.test[, 15]^2
R> xbeta.test <- xbeta.test + delta.test * (2 * trt.test - 1)
R> y.test <- xbeta.test + rnorm(10 * n.obs, sd = 2)
```

We then use the `predict()` function for objects returned by `fit.subgroup()` to obtain the estimated benefit scores for the test data:

```
R> bene.score.test <- predict(subgrp.model.eff, newx = x.test)
```

Finally we evaluate the subgroup-specific treatment effects on the test data based on the estimated subgroups and compare these values with confidence intervals from the bootstrap bias correction method. The effect of control among those recommended control is obtained by:

```
R> mean(y.test[bene.score.test <= 0 & trt.test == 0]) -
+    mean(y.test[bene.score.test <= 0 & trt.test == 1])

[1] 7.19437

R> quantile(validation.boot$boot.results[[1]][, 1], c(0.025, 0.975),
+    na.rm = TRUE)

     2.5%     97.5%
 1.824823 10.732421
```

The treatment effect among those recommended treatment is obtained by:

```
R> mean(y.test[bene.score.test > 0 & trt.test == 1]) -
+    mean(y.test[bene.score.test > 0 & trt.test == 0])

[1] 5.957166

R> quantile(validation.boot$boot.results[[1]][, 2], c(0.025, 0.975),
+    na.rm = TRUE)
```

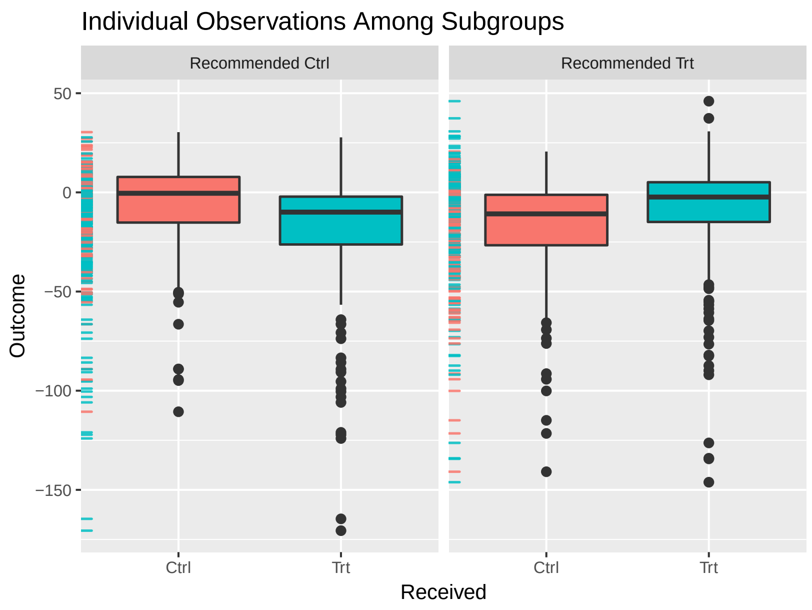

Figure 2: Individual outcomes within both the subgroup of patients whose benefit scores are positive and the subgroup of those whose benefit scores are negative.

```
     2.5%     97.5%
 3.266023 11.345697
```

We can see that the true values are contained within the bootstrap confidence intervals.

3.6. The `plot()` method and the `plotCompare()` function

The outcomes (or average outcomes) of patients within different subgroups can be plotted using the `plot()` function. In particular, this function plots patient outcomes by treatment statuses within each subgroup of patients. Boxplots of the outcomes can be plotted in addition to densities and interaction plot of the average outcomes within each of these groups. They can all be generated using code like the one below with resulting plots in Figures 2, 3, and 4:

```
R> plot(subgrp.model)
R> plot(subgrp.model, type = "density")
R> plot(subgrp.model, type = "interaction")
```

For objects of class '`subgroup_fitted`', specifying the argument `type = "conditional"` in the `plot()` function displays smoothed means of the outcomes conditional on each treatment group as a function of the benefit score. Thus, a meaningful subgroup will be revealed if the conditional means of the treated and untreated groups are not parallel in the benefit score. The conditional plot generated from the below code is in Figure 5.

```
R> plot(subgrp.model, type = "conditional")
```

Multiple models can be visually compared using the `plotCompare()` function, which offers the same plotting options as the `plot` method for '`subgroup_fitted`' objects.

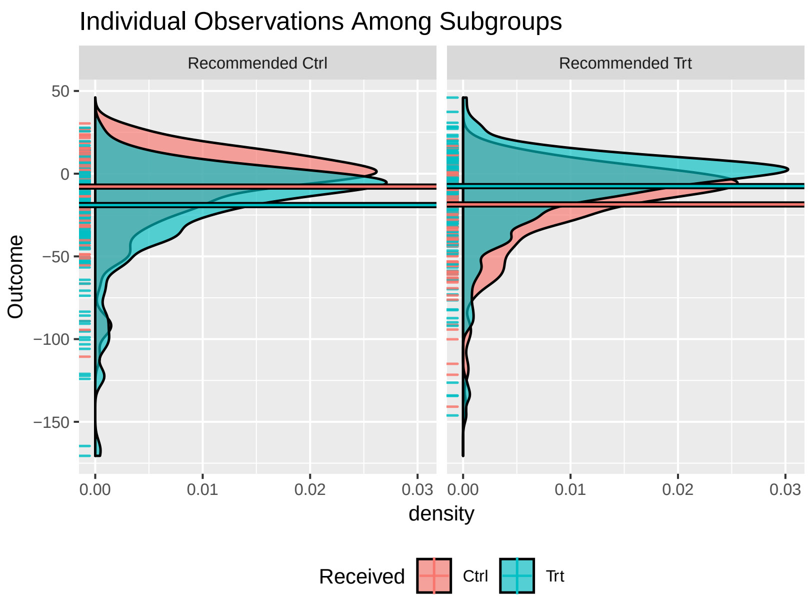

Figure 3: Density plots of individual outcome observations among the different subgroups.

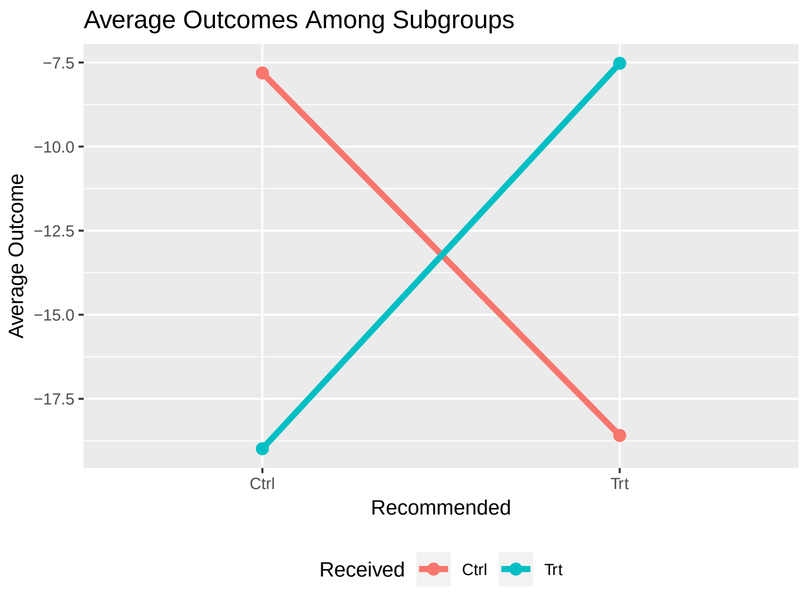

Figure 4: Interaction plot of the average outcome values within each subgroup by treatment status.

3.7. Efficiency augmentation

We now run the repeated training and testing splitting procedure on the model for continuous outcomes that did not utilize efficiency augmentation so we can compare with the efficiency-augmented model:

```
R> validation <- validate.subgroup(subgrp.model, B = 100,
+    method = "training_test_replication", train.fraction = 0.75)
```

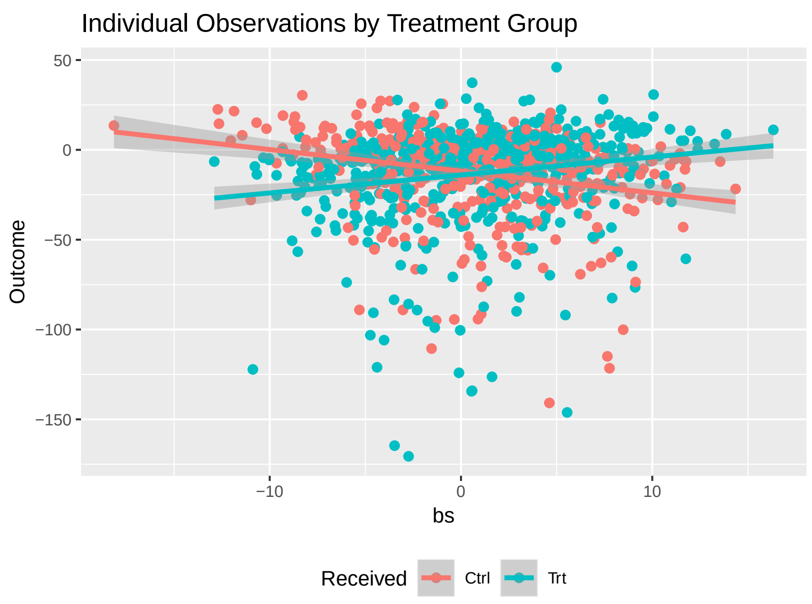

Figure 5: Individual observations of outcomes versus estimated benefit score by treatment status with smoothed mean lines by treatment arm.

```
R> validation
```

```
family:  gaussian
loss:    sq_loss_lasso
method:  a_learning

validation method:  training_test_replication
cutpoint:           0
replications:       100

benefit score: f(x),
Trt recom = Trt*I(f(x)>c)+Ctrl*I(f(x)<=c) where c is 'cutpoint'

Average Test Set Outcomes:
                              Recommended Ctrl
Received Ctrl  -10.83 (SE = 5.5725, n = 40.21)
Received Trt  -16.314 (SE = 4.9946, n = 59.98)
                                Recommended Trt
Received Ctrl -15.6022 (SE = 3.3228, n = 62.22)
Received Trt   -10.4702 (SE = 3.131, n = 87.59)

Treatment effects conditional on subgroups:
Est of E[Y|T=Ctrl,Recom=Ctrl]-E[Y|T=/=Ctrl,Recom=Ctrl]
                      5.5198 (SE = 8.5388, n = 100.19)
    Est of E[Y|T=Trt,Recom=Trt]-E[Y|T=/=Trt,Recom=Trt]
                        5.132 (SE = 5.019, n = 149.81)
```

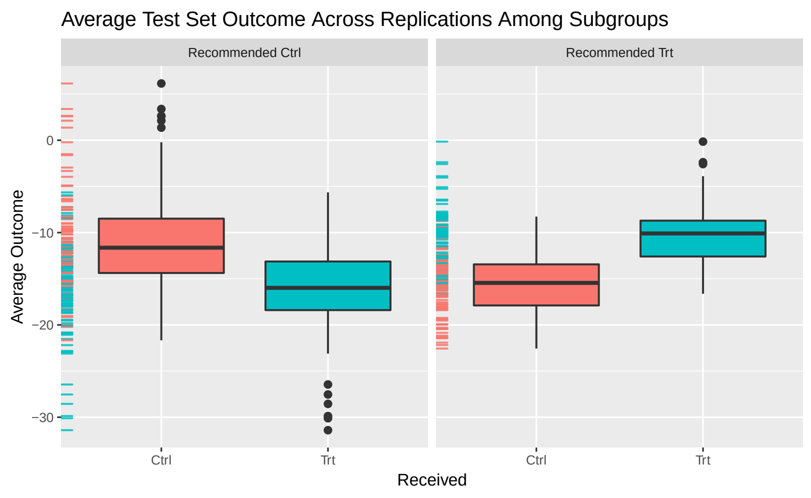

Figure 6: Values of average test set outcomes stratified by subgroups and treatment statuses across the training and testing replications.

```
Est of
E[Y|Trt received = Trt recom] - E[Y|Trt received =/= Trt recom]:
3.9346 (SE = 3.7614)
```

The results across the iterations for either the bootstrap of the training and testing partitioning procedure can be plotted using the `plot()` function similarly to how the `plot()` function can be used for fitted objects from `fit.subgroup()`. Similarly, boxplots, density plots, and interaction plots are all available through the `type` argument. Example code is below with resulting plot shown in Figure 6. For the sake of space, we do not show the density plot.

```
R> plot(validation)
R> plot(validation, type = "density")
```

Specifying the argument `type = "conditional"` plots the validation results conditional on different cutoff values for the benefit score as specified to `validate.subgroup()` via the `benefit.score.quantiles` argument. The resulting conditional plot generated by the below code is shown in Figure 7.

```
R> plot(validation, type = "conditional")
```

Multiple validated models can be visually compared using the `plotCompare()` function, which offers the same plotting options as the `plot` method for '`subgroup_validated`' objects. Here we compare the model fitted using `"sq_loss_lasso"` to the one fitted using `"sq_loss_lasso"` and efficiency augmentation. The resulting plot is shown in Figure 8.

```
R> plotCompare(validation, validation.eff)
```

From this comparison plot we can see that the efficiency-augmented model provides estimated subgroups that result in better overall outcomes when the recommended treatment is indeed the treatment received.

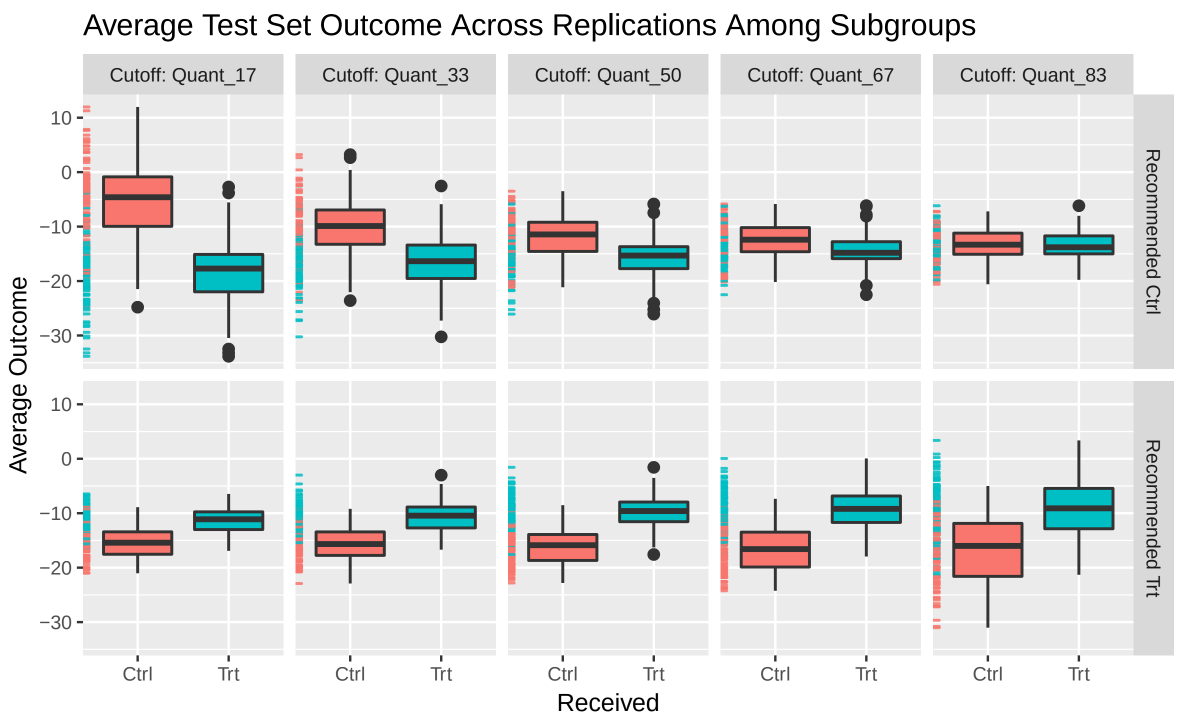

Figure 7: Values of average test set outcomes across the training and testing replications stratified by treatment statuses and subgroups as defined by different quantiles of the benefit score.

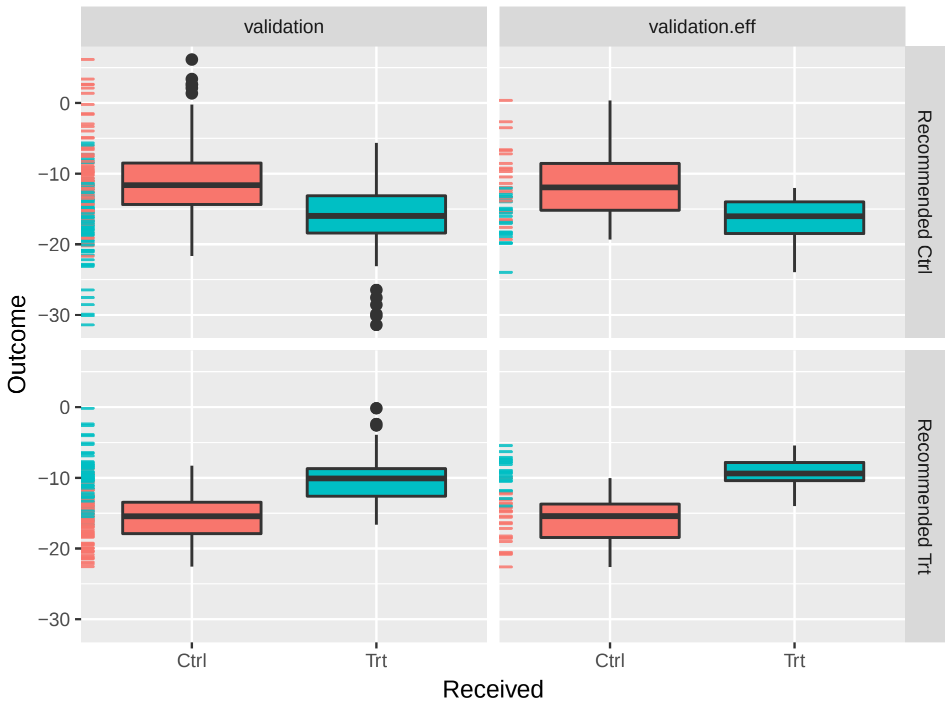

Figure 8: Comparison plot of the training and testing validation results for two different models.

3.8. Example with multi-category treatments

First we simulate data with three treatments. The treatment assignments will be based on covariates and hence mimic an observational setting with no unmeasured confounders. The term `delta1` below is the effect of treatment 1 relative to treatment 3 and `delta2` is defined similarly for treatment 2. The main effects have nonlinearities.

```
R> set.seed(123)
R> n.obs  <- 1000; n.vars <- 100
R> x <- matrix(rnorm(n.obs * n.vars, sd = 3), n.obs, n.vars)
R> xbetat_1 <- 0.1 + 0.5 * x[, 21] - 0.25 * x[, 25]
R> xbetat_2 <- 0.1 - 0.5 * x[, 11] + 0.25 * x[, 15]
R> trt.1.prob <- exp(xbetat_1) / (1 + exp(xbetat_1) + exp(xbetat_2))
R> trt.2.prob <- exp(xbetat_2) / (1 + exp(xbetat_1) + exp(xbetat_2))
R> trt.3.prob <- 1 - (trt.1.prob + trt.2.prob)
R> prob.mat <- cbind(trt.1.prob, trt.2.prob, trt.3.prob)
R> trt.mat <- apply(prob.mat, 1, function(rr) rmultinom(1, 1, prob = rr))
R> trt.num <- apply(trt.mat, 2, function(rr) which(rr == 1))
R> trt <- as.factor(paste0("Trt_", trt.num))
R> delta1 <- 2 * (0.5 + x[, 2] - 2 * x[, 3])
R> delta2 <- 0.5 + x[, 6] - 2 * x[, 5]
R> xbeta <- x[, 1] + x[, 11] - 2 * x[, 12]^2 + x[, 13] +
+    0.5 * x[, 15] ^ 2 + 2 * x[, 2] - 3 * x[, 5]
R> xbeta <- xbeta +  delta1 * ((trt.num == 1) - (trt.num == 3)) +
+    delta2 * ((trt.num == 2) - (trt.num == 3))
R> y <- xbeta + rnorm(n.obs, sd = 2)
```

We will use the `factor` version of the treatment status vector in our analysis, however, the integer values vector, i.e., `trt.num`, could be used as well.

```
R> trt[1:5]

[1] Trt_3 Trt_1 Trt_3 Trt_2 Trt_3
Levels: Trt_1 Trt_2 Trt_3

R> table(trt)

trt
Trt_1 Trt_2 Trt_3 
  368   359   273 
```

Then we construct a propensity score function that takes covariate information and the treatment statuses as input and generates a matrix of probabilities as output. Each row i of the output matrix represents an observation and each column j is the probability that the i-th patient received the j-th treatment. The treatment levels are ordered alphabetically (or numerically if the treatment assignment vector is a vector of integers). Our propensity score model in this example will be a multinomial logistic regression model with a lasso penalty for the probability of treatment assignments conditional on covariate information:

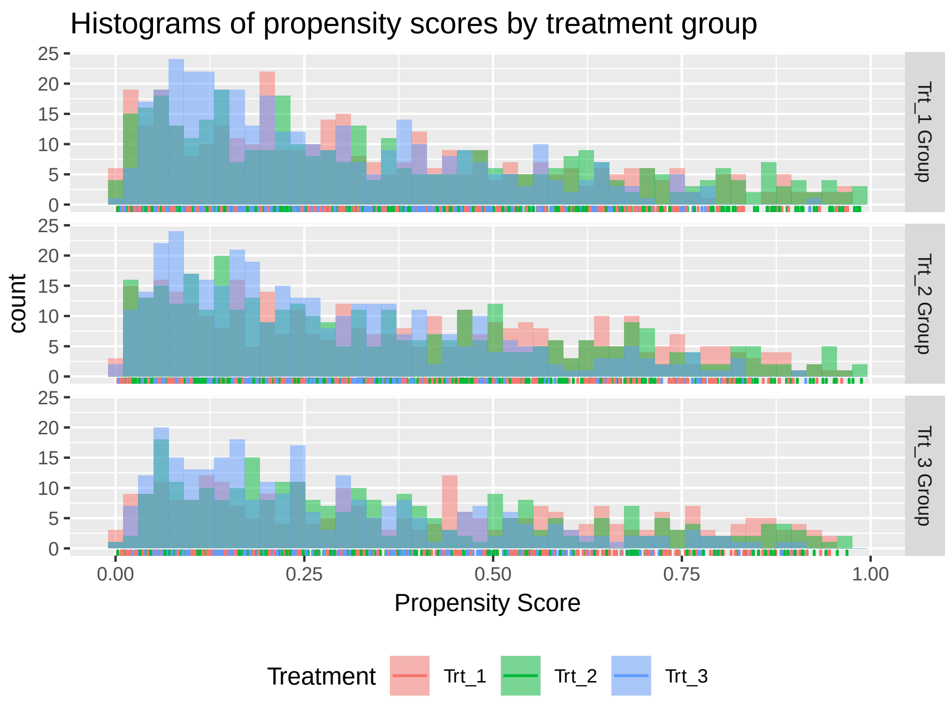

Figure 9: Propensity score overlap plot for multi-category treatment data.

```
R> propensity.multinom.lasso <- function(x, trt) {
+    if (!is.factor(trt)) trt <- as.factor(trt)
+    gfit <- cv.glmnet(y = trt, x = x, family = "multinomial")
+    propens <- drop(predict(gfit, newx = x,  type = "response",
+      s = "lambda.min"))
+    probs <- propens[, match(levels(trt), colnames(propens))]
+    probs
+  }
```

An important assumption for the propensity score is that $0 < \mathsf{P}(T_i = t|\mathbf{X}) < 1$ for all $\mathbf{X}$ and t. This assumption, often called the positivity assumption, is impossible to verify. However, in practice validity of the assumption can be assessed via a visualization of the empirical overlap of our estimated propensity scores to determine if there is any evidence of positivity violations. The `check.overlap()` function also allows us to visualize the overlap of our propensity scores for multi-category treatment applications. The following code results in the plot shown Figure 9.

```
R> check.overlap(x = x, trt = trt, propensity.multinom.lasso)
```

Each plot in Figure 9 is for a different treatment group, e.g., the plot in the first row of plots is the subset of patients who received treatment 1. There seems to be no obvious evidence against the positivity assumption.

As the outcome is continuous and there is a large number of covariates available for our construction of a benefit score, we will use the squared error loss and a lasso penalty. The model can be fit in the same manner as for the binary treatment setting, however only linear models and the weighting method are available. Here we can also specify the reference treatment (the treatment that the non-reference treatments are compared with by each benefit score). Here we specify that the reference treatment level is "Trt_3".

```
R> set.seed(123)
R> subgrp.multi <- fit.subgroup(x = x, y = y, trt = trt,
+    propensity.func = propensity.multinom.lasso, reference.trt = "Trt_3",
+    loss = "sq_loss_lasso")
R> summary(subgrp.multi)

family:    gaussian
loss:      sq_loss_lasso
method:    weighting
cutpoint:  0
propensity
function:  propensity.func

benefit score: f_Trt_1(x): Trt_1 vs Trt_3,  f_Trt_2(x): Trt_2 vs Trt_3
               f_Trt_3(x): 0
maxval = max(f_Trt_1(x), f_Trt_2(x))
which.max(maxval) = The trt level which maximizes maxval
Trt recom = which.max(maxval)*I(maxval > c) + Trt_3*I(maxval <= c)
where c is 'cutpoint'

Average Outcomes:
                Recommended Trt_1 Recommended Trt_2 Recommended Trt_3
Received Trt_1  -1.9711 (n = 170) -9.1388 (n = 131) -30.9531 (n = 67)
Received Trt_2 -16.9325 (n = 181)  2.2732 (n = 111) -31.8461 (n = 67)
Received Trt_3 -22.4541 (n = 127) -11.1916 (n = 86)  -2.7072 (n = 60)

Treatment effects conditional on subgroups:
Est of E[Y|T=Trt_1,Recom=Trt_1]-E[Y|T=/=Trt_1,Recom=Trt_1]
                                          17.6189 (n = 478)
Est of E[Y|T=Trt_2,Recom=Trt_2]-E[Y|T=/=Trt_2,Recom=Trt_2]
                                          12.4103 (n = 328)
Est of E[Y|T=Trt_3,Recom=Trt_3]-E[Y|T=/=Trt_3,Recom=Trt_3]
                                          28.6932 (n = 194)

NOTE: The above average outcomes are biased estimates of
      the expected outcomes conditional on subgroups.
      Use 'validate.subgroup()' to obtain unbiased estimates.

---------------------------------------------------

Benefit score 1 quantiles (f(X) for Trt_1 vs Trt_3):
     0%     25%     50%     75%    100%
-18.451  -2.903   2.216   6.885  22.101
```

```
Benefit score 2 quantiles (f(X) for Trt_2 vs Trt_3):
      0%      25%      50%      75%     100% 
-23.8125  -4.8801  -0.2818   4.7656  26.2459 

---------------------------------------------------

Summary of individual treatment effects: 
E[Y|T=trt, X] - E[Y|T=Trt_3, X]
where 'trt' is Trt_1 and Trt_2

 Trt_1-vs-Trt_3    Trt_2-vs-Trt_3   
 Min.   :-36.902   Min.   :-47.6249  
 1st Qu.: -5.807   1st Qu.: -9.7602  
 Median :  4.432   Median : -0.5636  
 Mean   :  4.415   Mean   : -0.4341  
 3rd Qu.: 13.771   3rd Qu.:  9.5311  
 Max.   : 44.202   Max.   : 52.4918  

---------------------------------------------------

13 out of 200 interactions selected in total by the lasso (cross validation
criterion).

The first estimate is the treatment main effect, which is always selected. 
Any other variables selected represent treatment-covariate interactions.


7 out of 100 variables selected for delta 1 by the lasso (cross validation
criterion).

                                      Trt_1     V2      V3    V10    V32
Estimates for delta (Trt_1 vs Trt_3) 2.0007 1.0344 -2.1732 0.1916 0.1272
                                         V61     V62     V79
Estimates for delta (Trt_1 vs Trt_3) -0.0834 -0.1028 -0.3649

6 out of 100 variables selected for delta 2 by the lasso (cross validation
criterion).

                                       Trt_2      V5    V11    V63     V80
Estimates for delta (Trt_2 vs Trt_3) -0.4728 -2.2508 0.0558 0.3312 -0.1906
                                         V92     V98
Estimates for delta (Trt_2 vs Trt_3) -0.0112 -0.7417
```

The `summary()` function now displays selected variables for each of the two benefit scores and shows the quantiles of each benefit score. We can also plot the empirical observations within the different subgroups using the `plot()` function, however now it is slightly more

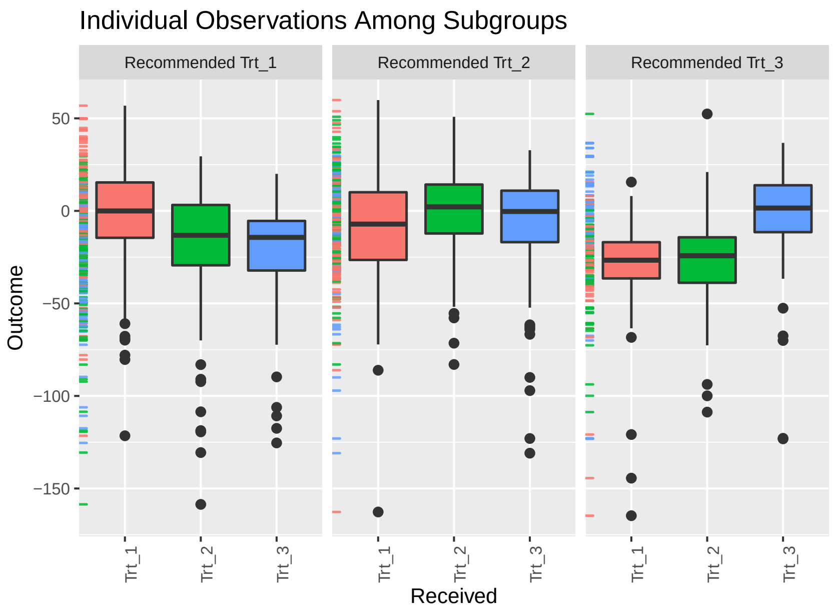

Figure 10: Individual outcome observations by treatment group and subgroup.

complicated. It appears that the average outcome is higher for those who received the level of the treatment they were recommended than those who received a different treatment than they were recommended. Also note that the `plot` method for '`subgroup_fitted`' objects returns a '`ggplot`' object (Wickham 2016; Wickham, Chang, Henry, Pedersen, Takahashi, Wilke, Woo, Yutani, and Dunnington 2021) and can thus be modified by the user. The below example yields Figure 10.

```
R> pl <- plot(subgrp.multi)
R> pl + theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

To obtain valid estimates of the subgroup-specific treatment effects, we perform the repeated training and testing resample procedure using the `validate.subgroup()` function:

```
R> set.seed(123)
R> validation.multi <- validate.subgroup(subgrp.multi, B = 100,
+    method = "training_test_replication", train.fraction = 0.5)
R> print(validation.multi, digits = 2, sample.pct = TRUE)

family:  gaussian
loss:    sq_loss_lasso
method:  weighting

validation method:  training_test_replication
cutpoint:           0
replications:       100

benefit score: f_Trt_1(x): Trt_1 vs Trt_3,  f_Trt_2(x): Trt_2 vs Trt_3
```

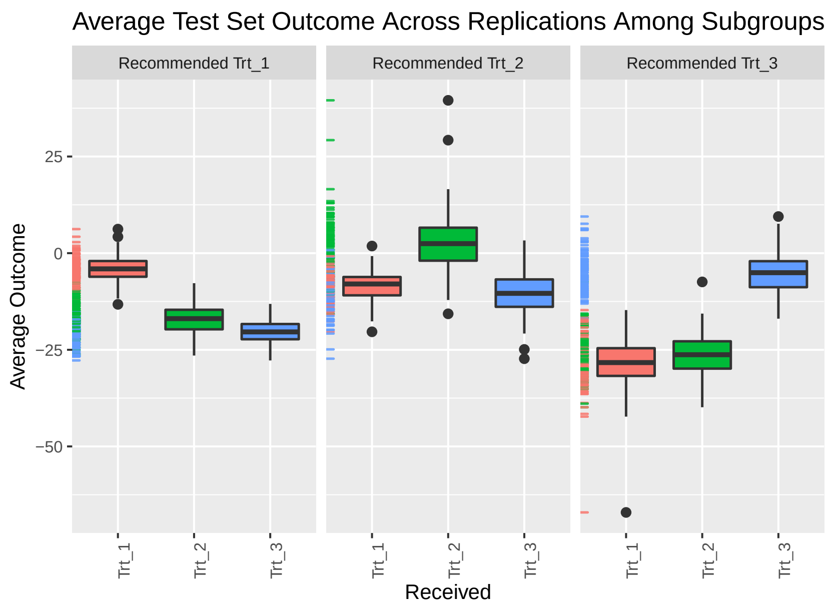

Figure 11: Validation results for multi-category treatment data.

```
               f_Trt_3(x): 0
maxval = max(f_Trt_1(x), f_Trt_2(x))
which.max(maxval) = The trt level which maximizes maxval
Trt recom = which.max(maxval)*I(maxval > c) + Trt_3*I(maxval <= c)
where c is 'cutpoint'

Average Test Set Outcomes:
                        Recommended Trt_1          Recommended Trt_2
Received Trt_1  -4.02 (SE = 3.29, 19.31%) -8.43 (SE = 3.83, 10.96%)
Received Trt_2 -16.98 (SE = 4.06, 20.25%)   2.49 (SE = 7.77, 9.02%)
Received Trt_3  -20.6 (SE = 3.16, 14.15%) -10.45 (SE = 5.47, 7.53%)
                       Recommended Trt_3
Received Trt_1 -28.34 (SE = 6.91, 6.53%)
Received Trt_2 -25.98 (SE = 5.47, 6.69%)
Received Trt_3  -4.89 (SE = 5.05, 5.57%)

Treatment effects conditional on subgroups:
Est of E[Y|T=Trt_1,Recom=Trt_1]-E[Y|T=/=Trt_1,Recom=Trt_1]
                                  14.72 (SE = 5.14, 53.7%)
Est of E[Y|T=Trt_2,Recom=Trt_2]-E[Y|T=/=Trt_2,Recom=Trt_2]
                                 11.93 (SE = 8.39, 27.52%)
Est of E[Y|T=Trt_3,Recom=Trt_3]-E[Y|T=/=Trt_3,Recom=Trt_3]
                                 22.24 (SE = 7.02, 18.78%)

Est of
E[Y|Trt received = Trt recom] - E[Y|Trt received =/= Trt recom]:
13.76 (SE = 3.04)
```

Setting the `sample.pct` argument above to `TRUE` prints out the average percent of all patients which are in each subgroup (as opposed to the average sample sizes). We can see that about 58% of patients were recommended treatment 1 and among those recommended treatment 1, we expect them to have larger outcomes if they actually receive treatment 1 as opposed to the other treatments. The estimated effects are positive within all three subgroups (meaning those recommended each of the different treatments have a positive benefit from receiving the treatment they are recommended as opposed to receiving any another treatments).

We can visualize the subgroup-specific treatment effects using `plot()` as usual with results shown in Figure 11:

```
R> plv <- plot(validation.multi)
R> plv + theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

4. Numerical comparisons

In this section we evaluate the finite sample performance of many different available methods in the **personalized** package and comparative methods available in other packages via a set of numerical studies. These comparative methods, outside the scope of the **personalized** package, are the residual-weighted learning (RWL) method of Zhou, Mayer-Hamblett, Khan, and Kosorok (2017) as implemented in the **DynTxRegime** package, the $\ell 1$-PLS approach of Qian and Murphy (2011, outcome-lasso), an outcome-modeling approach based on Bayesian additive regression trees (outcome-BART), and the model-based trees and forests approach of Seibold, Zeileis, and Hothorn (2016, 2017) implemented in the **model4you** package (Seibold, Zeileis, and Hothorn 2021). Comparison with more packages and methods would be ideal, however the majority of available packages either do not provide functions for prediction for new patients or are too computationally demanding. The methods from the **personalized** package utilized in this comparison are the A-learning method with the square loss and a lasso penalty (Sq-A); the weighting method with the square loss and a lasso penalty (Sq-W); the weighting method the flipped outcome weighted learning logistic loss and a lasso penalty, i.e., $\mathbf{M}(y, v) = |y| \log(1 + \exp\{-\text{sign}(y)v\})$, (FOWL-L-W); and the weighting method the flipped outcome weighted learning hinge loss, i.e., $\mathbf{M}(y, v) = |y| \max(0, 1 - \text{sign}(y)v)$, (FOWL-H-W). We additionally compare with loss-augmented versions of all of these aforementioned methods; the corresponding names have "-Aug" appended to the end, e.g., "Sq-W-Aug".

Covariates were generated as $\mathbf{X} = (X_1, \dots, X_p)^\top$, where $X_2, X_4, X_6, \dots, X_{40}$ are binary random variables with probability of success 0.25, and the remaining $p - 20$ elements of $\mathbf{X}$ are generated from a multivariate normal random variable with variance-covariance matrix 1 on the diagonal and $\rho^{|i-j|}$ for the element in the i-th row and j-th column with $\rho = 0.75$. Treatment statuses are generated from an observational study type setting with $\mathsf{P}(T = 1|\mathbf{X} = \mathbf{x}) = \text{expit}(\beta_{T0} + \boldsymbol{\beta}_T^\top \mathbf{x})$, where $\text{expit}(x) = 1/(1 + e^{-x})$ and the first 10 elements in $\boldsymbol{\beta}_T$ are generated from a uniform random variable on $[-0.5, -0.25] \cup [0.25, 0.5]$ and the rest are 0. The intercept β_{T0} is set such that on average 1/3 of observations would receive $T = 1$. The responses are generated from the following two models:

Model 1: $Y = \boldsymbol{\gamma}^\top \mathbf{X} + T\boldsymbol{\beta}^\top \mathbf{X} + \epsilon$

Model 2: $Y = \exp(0.5\boldsymbol{\gamma}^\top \mathbf{X}) - \exp(0.5\{\nu_1 X_1 X_2 + \nu_2 X_1 X_3 + \nu_3 X_2 X_3 + \nu_4 X_3 X_4 + \nu_5 X_5 X_6\}) + T\boldsymbol{\beta}^\top \mathbf{X} + \epsilon,$

where the first 10 elements in $\boldsymbol{\gamma}$ are generated from a uniform random variable on $[-c, -0.5c] \cup [0.5c, c]$ and the rest are 0, ν_i for $i = 1, \ldots, 5$ are generated from a uniform random variable on $[-c, -0.5c] \cup [0.5c, c]$, and 10 randomly chosen elements in $\boldsymbol{\beta}$ are generated from a uniform random variable on $[-1, -0.5] \cup [0.5, 1]$. For the large main covariate effects setting, $c = 4/3$ and for the moderate main covariate effects setting, $c = 2/3$.

For all methods that apply variable selection, the lasso is used and 10-fold cross-validation based on mean-squared error is used for selection of the tuning parameter. For all methods that require the use of a propensity score, the propensity score is created by fitting a binary logistic regression model with a lasso penalty. For all modeling options in the **personalized** package that do not use the hinge loss, the lasso is used for variable selection. For all methods in the **personalized** package that use outcome augmentation, a linear model `Y ~ x + x:trt` with the lasso is used to create the augmentation part. The treatment-covariate interactions are included in this function so that the main effects can be correctly specified. However, as the goal in subgroup identification is to estimate treatment-covariate interactions, the augmentation function we use averages over the predictions for `trt = 1` and `trt = -1`. We note that, under Model 2, this augmentation function is misspecified.

For the BART approach, we use the **BayesTree** package (Chipman and McCulloch 2016) with all covariates and the treatment indicator included and estimate the benefit score $\Delta(\mathbf{x})$ for each patient by evaluating the difference in predictions for `trt = 1` versus `trt = -1`. We use the default settings in the **BayesTree** package as the default settings are well-known to perform admirably (Chipman, George, McCulloch *et al.* 2010). Similarly, with the ℓ1-PLS approach, we fit a linear model with a lasso penalty for the outcome, including covariate main effects and treatment-covariate interactions. The benefit score is estimated in the same way as the BART approach. The augmentation function needed in creating the residuals for the RWL method is constructed in the same way as that used for the outcome augmentation of the **personalized** package.

The methods are evaluated by investigating the rank correlation between the true benefit score $\Delta(\mathbf{x})$ with its estimate $\widehat{f}(\mathbf{x})$ (or a monotone transformation of an estimate of $\Delta(\mathbf{x})$) on independent test sets of size 10000. Methods are also evaluated by their area under the receiver operating characteristic curves (AUC) with respect to the true underlying subgroups. The results are displayed in Figures 12 and 13, where "ME size: large" indicates the main effects are large, i.e., $c = 4/3$, and "ME size: small" indicates $c = 2/3$. The vertical dashed line separates methods from the **personalized** package and other methods. We did not include results for the outcome weighted learning losses, only the flipped versions of the outcome weighted learning losses as the flipped versions were uniformly better. Similarly, the tree-based version of the approach in **model4you** is not included, since the forest version of the method of **model4you** is uniformly better. However, the results are available in the supplementary material. Under Model 1, the ℓ1-PLS method is correctly specified thus can serve as the gold standard in terms of performance. However, under Model 2, the main effects are non-linear and the ℓ1-PLS model is incorrectly specified. Under Model 2, when the main effect size is small, the interaction effects dominate the main effects in size and outcome modeling approaches such as the ℓ1-PLS method are more robust to model misspecification than for large main effect sizes under Model 2.

We can see that augmentation with correct specification (Model 1) in the **personalized** package can boost performances although not necessarily so with incorrect specification (Model 2). Understandably, scenarios with larger main effects are associated with worse performance

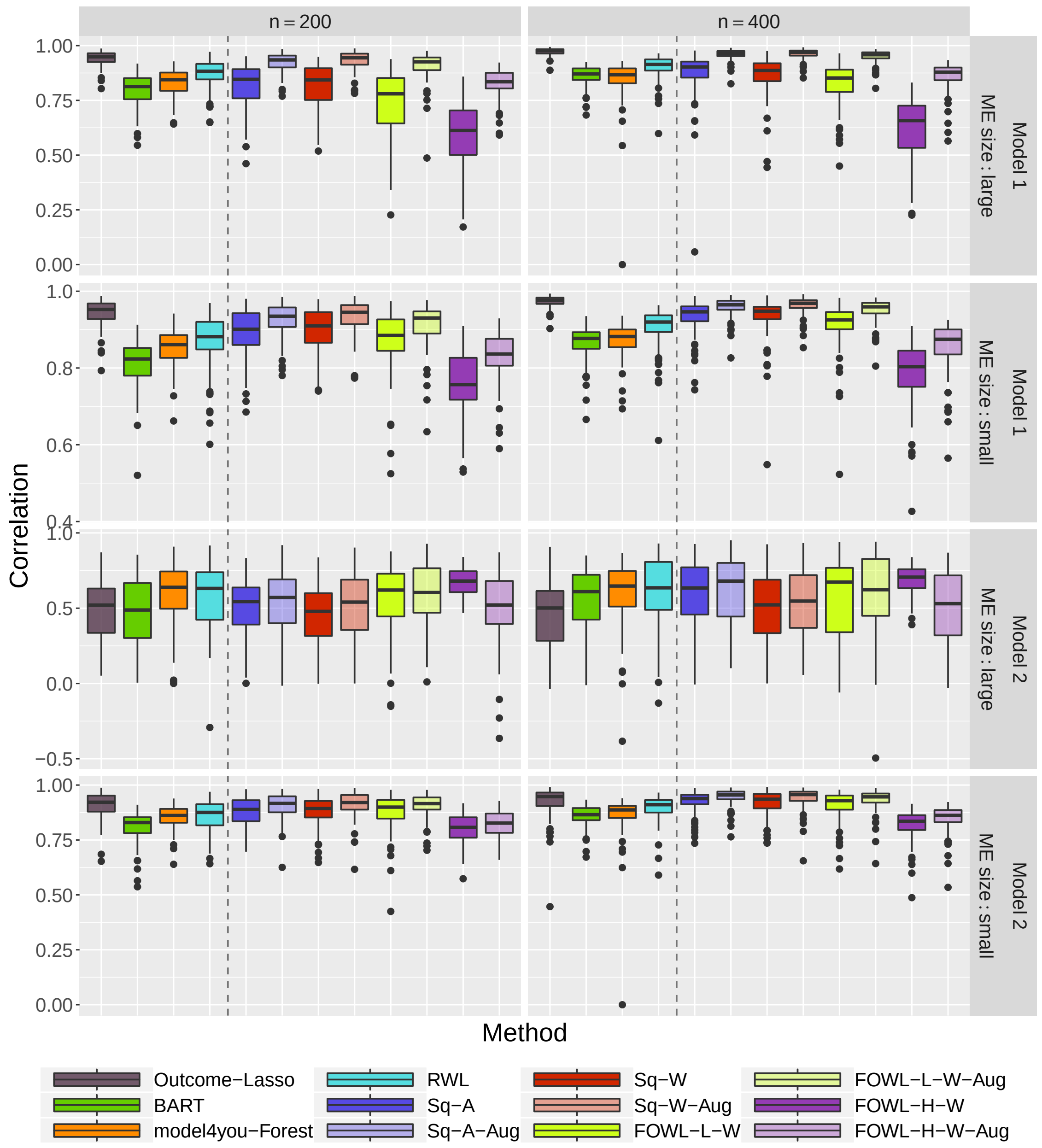

Figure 12: Correlations of the estimated benefit scores with the true benefit scores when the number of variables is 50. Results displayed are from 100 independent runs of the simulation.

for all methods. Of the two tree-based ensemble approaches, model4you-Forest tends to perform the best under most scenarios. Under Model 2 and large main effect sizes, the flipped outcome weighted learning approach with the hinge loss and no loss augmentation works very well and has the lowest variance. The flipped outcome weighted learning approach with loss augmentation with the logistic loss is never the best in any setting, however it is close to the best in all scenarios and is thus a reasonable choice in data scenarios where not much is known about the problem *a priori*.

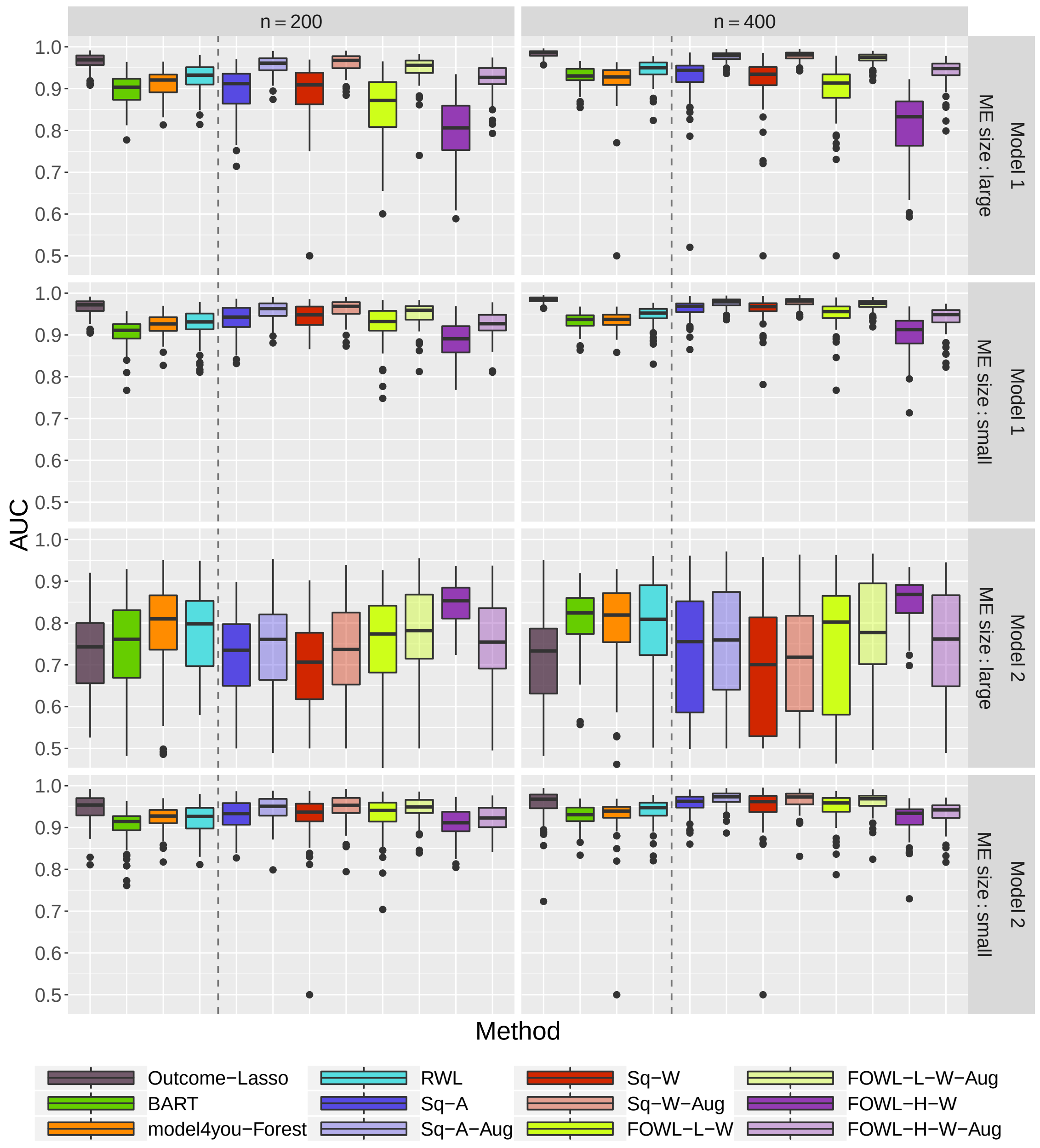

Figure 13: AUCs of the estimated benefit groups with respect to the true subgroups when the number of variables is 50. Results displayed are from 100 independent runs of the simulation.

5. Analysis of National Supported Work study

In this section we conduct a subgroup identification analysis for a study of the effectiveness of a training program designed to help under-served and under-employed workers gain the requisite skills for employment. The data came from the National Supported Work Study (LaLonde 1986). The outcome of interest is whether or not the earnings of individuals are greater in 1978 than in 1975 before the training program.

```
R> data("LaLonde", package = "personalized")
```

```
R> y <- LaLonde$outcome
```

The treatment assignment is whether each person had employment training or not. The object `data.x` is the set of covariates to be used in estimating the benefit score. Since it is a `data.frame`, we must use `model.matrix` to return a matrix object.

```
R> trt <- LaLonde$treat
R> x.varnames <- c("age", "educ", "black", "hisp", "white",  "marr",
+    "nodegr", "log.re75", "u75")
R> data.x <- LaLonde[, x.varnames]
R> x <- model.matrix(~ -1 + ., data = data.x)
```

The data come from a randomized controlled trial where patients were randomly assigned to either the supported work program or the control group. Even when the true propensity function is known, it is often more efficient to estimate it from the data. Hence we use the average number of those who were in the supported work program as the estimated propensity score.

```
R> const.propens <- function(x, trt) {
+    mean.trt <- mean(trt == "Trt")
+    rep(mean.trt, length(trt))
+  }
```

Here we fit a logistic regression-based estimator using the weighting method with the lasso penalty. We specify the `cv.glmnet()` argument `type.measure = "auc"` to specify the usage of area under the receiver operating characteristic curve (AUC) as the cross-validation metric for the determination of the lasso tuning parameter. We use the weighting method here only, since the results for the A-learning method were very similar.

```
R> set.seed(1)
R> subgrp_fit_w <- fit.subgroup(x = x, y = y, trt = trt,
+    loss = "logistic_loss_lasso", propensity.func = const.propens,
+    type.measure = "auc", nfolds = 10)
R> summary(subgrp_fit_w)

family:    binomial
loss:      logistic_loss_lasso
method:    weighting
cutpoint:  0
propensity
function:  propensity.func

benefit score: f(x),
Trt recom = Trt*I(f(x)>c)+Ctrl*I(f(x)<=c) where c is 'cutpoint'

Average Outcomes:
               Recommended Ctrl  Recommended Trt
Received Ctrl  0.7292 (n = 48) 0.5146 (n = 377)
Received Trt   0.5714 (n = 28) 0.6059 (n = 269)

Treatment effects conditional on subgroups:
Est of E[Y|T=Ctrl,Recom=Ctrl]-E[Y|T=/=Ctrl,Recom=Ctrl]
                                       0.1577 (n = 76)
    Est of E[Y|T=Trt,Recom=Trt]-E[Y|T=/=Trt,Recom=Trt]
                                      0.0914 (n = 646)

NOTE: The above average outcomes are biased estimates of
      the expected outcomes conditional on subgroups.
      Use 'validate.subgroup()' to obtain unbiased estimates.

---------------------------------------------------

Benefit score quantiles (f(X) for Trt vs Ctrl):
     0%     25%     50%     75%    100%
-0.2034  0.1334  0.1334  0.1334  0.3158

---------------------------------------------------

Summary of individual treatment effects:
E[Y|T=Trt, X] - E[Y|T=Ctrl, X]

    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.10133  0.06658  0.06658  0.06351  0.06658  0.15661

---------------------------------------------------

2 out of 10 interactions selected in total by the lasso (cross validation
criterion).

The first estimate is the treatment main effect, which is always selected.
Any other variables selected represent treatment-covariate interactions.

            Trt hispYes marrYes
Estimate 0.1334 -0.3367  0.1825
```

To evaluate the impact of the estimated subgroups, we randomly split the data into a training portion (80%) and a testing portion (the remaining 20%), fit a benefit score model on the training portion, and evaluate the treatment effects within the estimated subgroups on the testing portion with 500 replications. This allows us to determine whether using our benefit score to make treatment decisions for patients will result in better outcomes.

```
R> val_subgrp_w <- validate.subgroup(subgrp_fit_w, B = 500,
+    method = "training", train.fraction = 0.80)
R> print(val_subgrp_w, digits = 4, sample.pct = TRUE)

family:  binomial
```

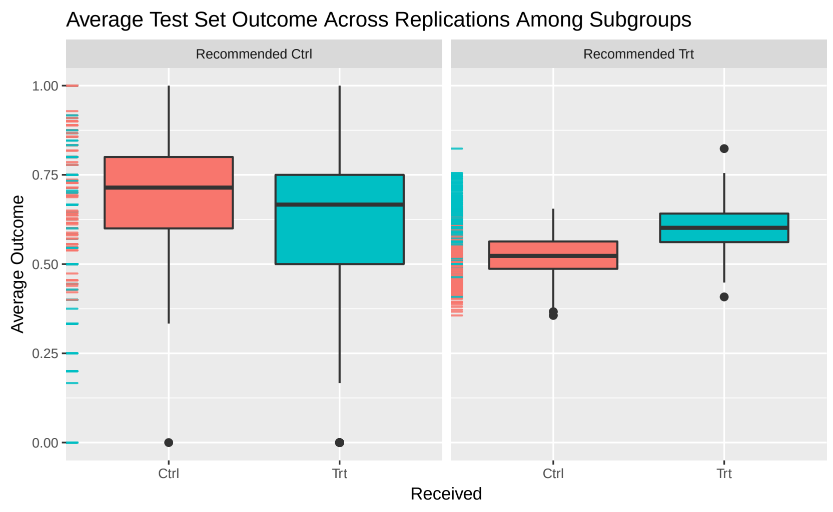

Figure 14: Average test set outcomes across training and testing replications stratified by subgroup and treatment status.

```
loss:    logistic_loss_lasso
method:  weighting

validation method:  training_test_replication
cutpoint:           0
replications:       500

benefit score: f(x),
Trt recom = Trt*I(f(x)>c)+Ctrl*I(f(x)<=c) where c is 'cutpoint'

Average Test Set Outcomes:
                           Recommended Ctrl                Recommended Trt
Received Ctrl 0.7058 (SE = 0.1453, 5.4883%)  0.521 (SE = 0.0538, 53.4566%)
Received Trt  0.6341 (SE = 0.2012, 3.5476%) 0.6017 (SE = 0.0611, 37.5076%)

Treatment effects conditional on subgroups:
Est of E[Y|T=Ctrl,Recom=Ctrl]-E[Y|T=/=Ctrl,Recom=Ctrl]
                         0.0717 (SE = 0.2359, 9.0359%)
    Est of E[Y|T=Trt,Recom=Trt]-E[Y|T=/=Trt,Recom=Trt]
                        0.0807 (SE = 0.0837, 90.9641%)

Est of
E[Y|Trt received = Trt recom] - E[Y|Trt received =/= Trt recom]:
0.0757 (SE = 0.0779)
```

The results over the 500 training and testing replications are displayed in Figure 14.

```
R> plot(val_subgrp_w)
```

The `plotCompare()` function allows us to inspect the treatment effects within subgroups

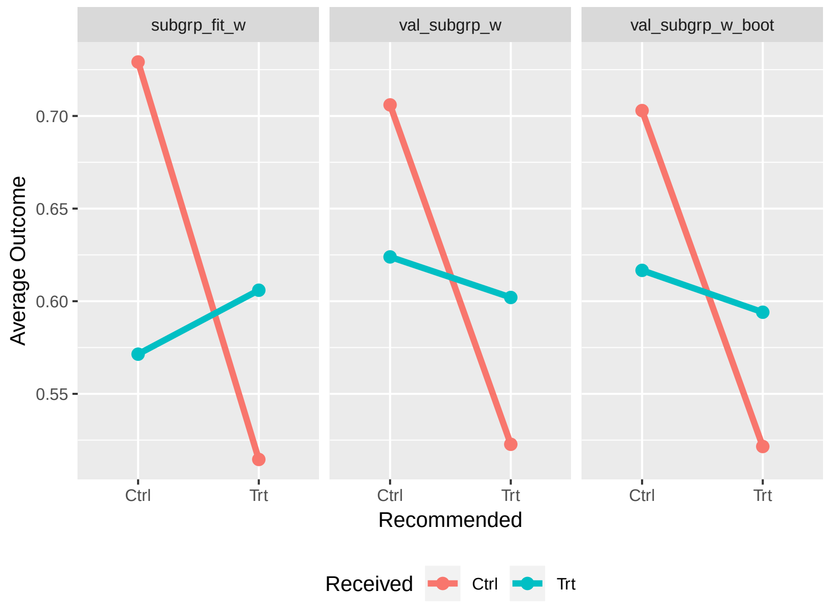

Figure 15: Interaction plot showing the difference in the empirical averages of outcomes on the training data compared with the average test set results across the training and testing replications.

on the testing datasets across all repetitions of the validation procedure in comparison with the estimated subgroup treatment effects using the training data. This comparison, with an additional comparison with the estimates generated from the bootstrap bias correction approach, is displayed in Figure 15. The estimates of the subgroup treatment effects based on the training data is likely overly-optimistic. However, the training/testing procedure allows us to correct a possible overfitting. We can see that among those who are recommended to receive the employment training, those who actually received the employment training were more likely to have a higher salary than those who did not receive the training. On the other hand, among those who were not recommended the training, those who did not receive the training may have a slightly higher salary. However there seem to be much more variation. Indeed the estimate of the benefit of employment training is attenuated for the validation-based estimates compared with the biased empirical estimates. Across the replications approximately 91% of the samples were recommended to receive employment training.

```
R> val_subgrp_w_boot <- validate.subgroup(subgrp_fit_w, B = 500,
+    method = "boot")
R> print(val_subgrp_w_boot, digits = 4, sample.pct = TRUE)

family:  binomial
loss:    logistic_loss_lasso
method:  weighting

validation method:  boot_bias_correction
cutpoint:           0
```

```
replications:       500

benefit score: f(x),
Trt recom = Trt*I(f(x)>c)+Ctrl*I(f(x)<=c) where c is 'cutpoint'

Average Bootstrap Bias-Corrected Outcomes:
                           Recommended Ctrl                Recommended Trt
Received Ctrl 0.6995 (SE = 0.0538, 6.5224%) 0.5246 (SE = 0.0298, 52.3299%)
Received Trt  0.6203 (SE = 0.0668, 3.9615%) 0.5922 (SE = 0.0293, 37.1861%)

Treatment effects conditional on subgroups:
Est of E[Y|T=Ctrl,Recom=Ctrl]-E[Y|T=/=Ctrl,Recom=Ctrl]
                        0.0796 (SE = 0.0805, 10.4839%)
    Est of E[Y|T=Trt,Recom=Trt]-E[Y|T=/=Trt,Recom=Trt]
                         0.0676 (SE = 0.039, 89.5161%)

Est of
E[Y|Trt received = Trt recom] - E[Y|Trt received =/= Trt recom]:
0.0631 (SE = 0.0373)
```

The bootstrap bias correction approach yields very similar estimates of the subgroup-conditional treatment effects. However, the bootstrap bias correction approach has smaller standard errors. This aligns with the findings of Foster *et al.* (2011), who noted that cross-validation type approaches lead to excessively high standard errors.

```
R> plotCompare(subgrp_fit_w, val_subgrp_w, val_subgrp_w_boot, type = "int")
```

6. Discussion

The **personalized** package provides simple-to-use routines for subgroup identification and personalized medicine via the general subgroup identification framework of Chen *et al.* (2017). The methods available under this framework cover a wide variety of models, outcomes, and loss functions all under a unified code structure. The underlying code is also designed to incorporate new models and loss functions that fall under the purview of the framework of Chen *et al.* (2017). The **personalized** package offers multiple methods for validating the impact of estimated subgroups and various ways of visualizing and inspecting the estimated subgroups and the resulting subgroup treatment effects. We hope to make subgroup identification and personalized medicine available to more statisticians and practitioners by making the entire subgroup identification analysis process as simple, understandable, and general as possible.

Acknowledgments

Research reported in this article was partially funded through a Patient-Centered Outcomes Research Institute (PCORI) Award (ME-1409-21219). The views in this publication are solely the responsibility of the authors and do not necessarily represent the views of the PCORI, its Board of Governors or Methodology Committee.

References

Athey S, Imbens G (2016). “Recursive Partitioning for Heterogeneous Causal Effects.” *Proceedings of the National Academy of Sciences of the United States of America*, **113**(27), 7353–7360. doi:10.1073/pnas.1510489113.

Bartlett PL, Jordan MI, McAuliffe JD (2006). “Convexity, Classification, and Risk Bounds.” *Journal of the American Statistical Association*, **101**(473), 138–156. doi:10.1198/016214505000000907.

Caliendo M, Kopeinig S (2008). “Some Practical Guidance for the Implementation of Propensity Score Matching.” *Journal of Economic Surveys*, **22**(1), 31–72. doi:10.1111/j.1467-6419.2007.00527.x.

Chen PY, Tsiatis AA (2001). “Causal Inference on the Difference of the Restricted Mean Lifetime between Two Groups.” *Biometrics*, **57**(4), 1030–1038. doi:10.1111/j.0006-341x.2001.01030.x.

Chen S, Tian L, Cai T, Yu M (2017). “A General Statistical Framework for Subgroup Identification and Comparative Treatment Scoring.” *Biometrics*, **73**(4), 1199–1209. doi:10.1111/biom.12676.

Chipman H, McCulloch R (2016). ***BayesTree**: Bayesian Additive Regression Trees.* R package version 0.3-1.4, URL https://CRAN.R-project.org/package=BayesTree.

Chipman HA, George EI, McCulloch RE, *et al.* (2010). “BART: Bayesian Additive Regression Trees.” *The Annals of Applied Statistics*, **4**(1), 266–298. doi:10.1214/09-aoas285.

Ciarleglio A, Petkova E, Ogden RT, Tarpey T (2015). “Treatment Decisions Based on Scalar and Functional Baseline Covariates.” *Biometrics*, **71**(4), 884–894. doi:10.1111/biom.12346.

Crump RK, Hotz VJ, Imbens GW, Mitnik OA (2009). “Dealing with Limited Overlap in Estimation of Average Treatment Effects.” *Biometrika*, **96**(1), 187–199. doi:10.1093/biomet/asn055.

der Elst WV, Alonso A, Molenberghs G (2020). ***EffectTreat**: Prediction of Therapeutic Success.* R package version 1.1, URL https://CRAN.R-project.org/package=EffectTreat.

Dusseldorp E, Doove L, Van de Put J, Mechelen IV, Claramunt Gonzalez J (2020). ***quint**: Qualitative Interaction Trees.* R package version 2.1.0, URL https://CRAN.R-project.org/package=quint.

Egami N, Ratkovic M, Imai K (2019). ***FindIt**: Finding Heterogeneous Treatment Effects.* R package version 1.2.0, URL https://CRAN.R-project.org/package=FindIt.

Foster JC, Taylor JMG, Ruberg SJ (2011). “Subgroup Identification from Randomized Clinical Trial Data.” *Statistics in Medicine*, **30**(24), 2867–2880. doi:10.1002/sim.4322.

Friedman J, Hastie T, Tibshirani R (2010). “Regularization Paths for Generalized Linear Models via Coordinate Descent.” *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.

Friedman J, Hastie T, Tibshirani R, Narasimhan B, Tay K, Simon N (2021). ***glmnet**: Lasso and Elastic-Net Regularized Generalized Linear Models.* R package version 4.1-1, URL `https://CRAN.R-project.org/package=glmnet`.

Garrido MM, Kelley AS, Paris J, Roza K, Meier DE, Morrison RS, Aldridge MD (2014). "Methods for Constructing and Assessing Propensity Scores." *Health Services Research*, **49**(5), 1701–1720. `doi:10.1111/1475-6773.12182`.

Greenwell B, Boehmke B, Cunningham J, GBM Developers (2020). ***gbm**: Generalized Boosted Regression Models.* R package version 2.1.8, URL `https://CRAN.R-project.org/package=gbm`.

Harrell FE, Lee KL, Mark DB (1996). "Multivariable Prognostic Models: Issues in Developing Models, Evaluating Assumptions and Adequacy, and Measuring and Reducing Errors." *Statistics in Medicine*, **15**(4), 361–387. `doi:10.1002/(sici)1097-0258(19960229)15:4<361::aid-sim168>3.0.co;2-4`.

Holloway ST, Laber EB, Linn KA, Zhang B, Davidian M, Tsiatis AA (2020). ***DynTxRegime**: Methods for Estimating Optimal Dynamic Treatment Regimes.* R package version 4.9, URL `https://CRAN.R-project.org/package=DynTxRegime`.

Hommel G (1988). "A Stagewise Rejective Multiple Test Procedure Based on a Modified Bonferroni Test." *Biometrika*, **75**(2), 383–386. `doi:10.1093/biomet/75.2.383`.

Huang X, Sun Y, Chatterjee S, Trow P (2017). ***SubgrpID**: Patient Subgroup Identification for Clinical Drug Development.* R package version 0.11, URL `https://CRAN.R-project.org/package=SubgrpID`.

Huling J (2021). ***personalized**: Estimation and Validation Methods for Subgroup Identification and Personalized Medicine.* R package version 0.2.6, URL `https://CRAN.R-project.org/package=personalized`.

Imbens GW, Rubin DB (2015). *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction.* Cambridge University Press. `doi:10.1017/cbo9781139025751`.

Irwin J (1949). "The Standard Error of an Estimate of Expectation of Life, with Special Reference to Expectation of Tumourless Life in Experiments with Mice." *The Journal of Hygiene*, **47**(2), 188. `doi:10.1017/s0022172400014443`.

Jeng XJ, Lu W, Peng H, *et al.* (2018). "High-Dimensional Inference for Personalized Treatment Decision." *Electronic Journal of Statistics*, **12**(1), 2074–2089. `doi:10.1214/18-ejs1439`.

Karatzoglou A, Smola A, Hornik K (2019). ***kernlab**: Kernel-Based Machine Learning Lab.* R package version 0.9-29, URL `https://CRAN.R-project.org/package=kernlab`.

Karatzoglou A, Smola A, Hornik K, Zeileis A (2004). "**kernlab** – An S4 Package for Kernel Methods in R." *Journal of Statistical Software*, **11**(9), 1–20. `doi:10.18637/jss.v011.i09`.

LaLonde RJ (1986). "Evaluating the Econometric Evaluations of Training Programs with Experimental Data." *The American Economic Review*, **76**(4), 604–620.

Lu W, Zhang HH, Zeng D (2013). "Variable Selection for Optimal Treatment Decision." *Statistical Methods in Medical Research*, **22**(5), 493–504. doi:10.1177/0962280211428383.

McCaffrey DF, Griffin BA, Almirall D, Slaughter ME, Ramchand R, Burgette LF (2013). "A Tutorial on Propensity Score Estimation for Multiple Treatments Using Generalized Boosted Models." *Statistics in Medicine*, **32**(19), 3388–3414. doi:10.1002/sim.5753.

Qian M, Murphy SA (2011). "Performance Guarantees for Individualized Treatment Rules." *The Annals of Statistics*, **39**(2), 1180–1210. doi:10.1214/10-aos864.

Qiu X, Zeng D, Wang Y (2018). "Estimation and Evaluation of Linear Individualized Treatment Rules to Guarantee Performance." *Biometrics*, **74**(2), 517–528. doi:10.1111/biom.12773.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Riviere MK (2021). ***SIDES**: Subgroup Identification Based on Differential Effect Search*. R package version 1.16, URL https://CRAN.R-project.org/package=SIDES.

Rosenbaum PR, Rubin DB (1983). "The Central Role of the Propensity Score in Observational Studies for Causal Effects." *Biometrika*, **70**(1), 41–55. doi:10.1093/biomet/70.1.41.

Rubin DB (2005). "Causal Inference Using Potential Outcomes: Design, Modeling, Decisions." *Journal of the American Statistical Association*, **100**(469), 322–331. doi:10.1198/016214504000001880.

Seibold H, Zeileis A, Hothorn T (2016). "Model-Based Recursive Partitioning for Subgroup Analyses." *The International Journal of Biostatistics*, **12**(1), 45–63. doi:10.1515/ijb-2015-0032.

Seibold H, Zeileis A, Hothorn T (2017). "Individual Treatment Effect Prediction for Amyotrophic Lateral Sclerosis Patients." *Statistical Methods in Medical Research*, **27**(10), 3104–3125. doi:10.1177/0962280217693034.

Seibold H, Zeileis A, Hothorn T (2021). ***model4you**: Stratified and Personalised Models Based on Model-Based Trees and Forests*. R package version 0.9-7, URL https://CRAN.R-project.org/package=model4you.

Shi C, Fan A, Song R, Lu W, *et al.* (2018). "High-Dimensional A-Learning for Optimal Dynamic Treatment Regimes." *The Annals of Statistics*, **46**(3), 925–957. doi:10.1214/17-aos1570.

Shi C, Song R, Lu W (2016). "Robust Learning for Optimal Treatment Decision with NP-Dimensionality." *Electronic Journal of Statistics*, **10**(2), 2894–2921. doi:10.1214/16-ejs1178.

Tewari A, Bartlett PL (2007). "On the Consistency of Multiclass Classification Methods." *Journal of Machine Learning Research*, **8**(May), 1007–1025.

Therneau TM (2021). ***survival**: Survival Analysis*. R package version 3.2-11, URL https://CRAN.R-project.org/package=survival.

Tian L, Alizadeh AA, Gentles AJ, Tibshirani R (2014). "A Simple Method for Estimating Interactions between a Treatment and a Large Number of Covariates." *Journal of the American Statistical Association*, **109**(508), 1517–1532. doi:10.1080/01621459.2014.951443.

Van Klaveren D, Steyerberg EW, Serruys PW, Kent DM (2018). "The Proposed 'Concordance-Statistic for Benefit' Provided a Useful Metric When Modeling Heterogeneous Treatment Effects." *Journal of Clinical Epidemiology*, **94**, 59–68. doi:10.1016/j.jclinepi.2017.10.021.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis.* 2nd edition. Springer-Verlag, New York. doi:10.1007/978-3-319-24277-4.

Wickham H, Chang W, Henry L, Pedersen TL, Takahashi K, Wilke C, Woo K, Yutani H, Dunnington D (2021). ***ggplot2**: Create Elegant Data Visualisations Using the Grammar of Graphics.* R package version 3.3.3, URL https://CRAN.R-project.org/package=ggplot2.

Wood S (2019). ***mgcv**: Mixed GAM Computation Vehicle with GCV/AIC/REML Smoothness Estimation.* R package version 1.8-31, URL https://CRAN.R-project.org/package=mgcv.

Wood SN (2017). *Generalized Additive Models: An Introduction with R.* 2nd edition. Chapman & Hall/CRC.

Xu Y, Yu M, Zhao YQ, Li Q, Wang S, Shao J (2015). "Regularized Outcome Weighted Subgroup Identification for Differential Treatment Effects." *Biometrics*, **71**(3), 645–653. doi:10.1111/biom.12322.

Zhang C, Liu Y (2014). "Multicategory Angle-Based Large-Margin Classification." *Biometrika*, **101**(3), 625–640. doi:10.1093/biomet/asu017.

Zhao H, Tsiatis AA (1997). "A Consistent Estimator for the Distribution of Quality Adjusted Survival Time." *Biometrika*, **84**(2), 339–348. doi:10.1093/biomet/84.2.339.

Zhao H, Tsiatis AA (1999). "Efficient Estimation of the Distribution of Quality-Adjusted Survival Time." *Biometrics*, **55**(4), 1101–1107. doi:10.1111/j.0006-341x.1999.01101.x.

Zhao Q, Small DS, Ertefaie A (2017). "Selective Inference for Effect Modification via the Lasso." arXiv:1705.08020 [math.ST], URL http://arxiv.org/abs/1705.08020.

Zhao Y, Zeng D, Rush AJ, Kosorok MR (2012). "Estimating Individualized Treatment Rules Using Outcome Weighted Learning." *Journal of the American Statistical Association*, **107**(499), 1106–1118. doi:10.1080/01621459.2012.695674.

Zhou X, Mayer-Hamblett N, Khan U, Kosorok MR (2017). "Residual Weighted Learning for Estimating Individualized Treatment Rules." *Journal of the American Statistical Association*, **112**(517), 169–187. doi:10.1080/01621459.2015.1093947.

Zou H, Zhu J, Hastie T (2008). "New Multicategory Boosting Algorithms Based on Multicategory Fisher-Consistent Losses." *The Annals of Applied Statistics*, **2**(4), 1290–1306. doi:10.1214/08-aoas198.

Affiliation:

Jared D. Huling
University of Minnesota
420 Delaware St. SE
Minneapolis, Minnesota 55455, United States of America
E-mail: huling@umn.edu
URL: http://jaredhuling.org/

Menggang Yu
University of Wisconsin-Madison
600 Highland Ave.
Madison, WI 53792, United States of America
E-mail: meyu@biostat.wisc.edu

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
May 2021, Volume 98, Issue 5 *Submitted:* 2018-04-14
doi:10.18637/jss.v098.i05 *Accepted:* 2020-12-28