A Bayesian Machine Learning Approach for Optimizing Dynamic Treatment Regimes

Thomas A. Murray[*,†] (tamurray@mdanderson.org),
Ying Yuan[*,*] (yyuan@mdanderson.org), and
Peter F. Thall[†] (rex@mdanderson.org)
Department of Biostatistics, MD Anderson Cancer Center

April 21, 2017

Abstract

Medical therapy often consists of multiple stages, with a treatment chosen by the physician at each stage based on the patient's history of treatments and clinical outcomes. These decisions can be formalized as a dynamic treatment regime. This paper describes a new approach for optimizing dynamic treatment regimes that bridges the gap between Bayesian inference and existing approaches, like Q-learning. The proposed approach fits a series of Bayesian regression models, one for each stage, in reverse sequential order. Each model uses as a response variable the remaining payoff assuming optimal actions are taken at subsequent stages, and as covariates the current history and relevant actions at that stage. The key difficulty is that the optimal decision rules at subsequent stages are unknown, and even if these decision rules were known the relevant response variables may be counterfactual. However, posterior distributions can be derived from the previously fitted regression models for the optimal decision rules and the counterfactual response variables under a particular set of rules. The proposed approach averages over these posterior distributions when fitting each regression model. An efficient sampling algorithm for estimation is presented, along with simulation studies that compare the proposed approach with Q-learning.

Keywords: Approximate Dynamic Programming; Backward Induction; Bayesian Additive Regression Trees; Gibbs Sampling; Potential Outcomes

[*]Yuan's and Murray's research was partially supported by Award Number R01-CA154591 from the National Cancer Institute.

[†]The work of the first three authors was partially funded by NIH/NCI grant 5-R01-CA083932.

1 Introduction

In medical practice, therapy often consists of a series of treatments assigned in multiple stages. The physician chooses each treatment adaptively based on the patient's history of treatments and clinical outcomes at that stage. For example, oncologists typically choose an initial (frontline) treatment based on disease severity and other prognostic covariates. At the times of one or more subsequent disease progressions, a salvage treatment is chosen based on the patient's prior treatments and responses. Most statistical methods for treatment evaluation ignore this process and focus on evaluating the treatments given at a single stage, which most often is either frontline or first salvage. Because this strategy does not reflect the actual therapeutic process, it is of limited use to practicing physicians. In the above oncology example, if survival is the main outcome of interest then evaluating only frontline treatment options ignores the effects of salvage treatments on the patient's survival. Although it is well known that this myopic approach is likely to lead to sub-optimal therapeutic decisions (Lavori and Dawson, 2004; Chakraborty, 2011), it is commonly used because evaluating treatments given at a single stage is much simpler than evaluating a series of treatments given at multiple stages.

Multi-stage treatment decisions can be formalized as a dynamic treatment regime (DTR), which is a set of decision rules, one for each stage, that stipulate which treatment to assign (or action to take) based on the patient's history at that stage. Prior to the seminal papers by Murphy (2003) and Robins (2004), there was a dearth of statistical methods for optimizing DTRs. In recent years, many approaches for defining, estimating, and optimizing DTRs have been proposed. These include Moodie et al. (2007), Zhao et al. (2009), Chakraborty et al. (2010), Arjas and Saarela (2010), Wang et al. (2012), Zajonc (2012), Zhang et al. (2013), Huang et al. (2015), Zhao et al. (2015) and Xu et al. (2016). We provide a brief survey of existing methods for optimizing DTRs below, in Section 2.

This paper describes a new approach that bridges the present gap between Bayesian inference and approximate dynamic programming methods from machine learning, like Q-learning

(Watkins, 1989; Moodie et al., 2007). Our proposed approach, which we call Bayesian Machine Learning (BML), is a novel approximate dynamic programming approach that fits a series of Bayesian regression models, one for each stage, in reverse sequential order. Each model uses as a response variable the remaining payoff assuming optimal actions (or treatments) are taken at subsequent stages, and as covariates the current history and relevant actions at that stage. One difficulty in fitting a regression model for a non-terminal stage is that the optimal decision rule at each subsequent stage is unknown. Another difficulty is that, even if the optimal decision rule at each subsequent stage was known, the relevant response variable is counterfactual for a sample observation that did not follow the optimal action at each subsequent stage. The proposed approach relies on the critical observation that posterior distributions can be derived from the previously fitted regression models for the unknown optimal decision rule at each subsequent stage, and for the counterfactual response variables that arise for a particular set of decision rules at subsequent stages. One distinguishing feature of our proposed BML approach is that it treats the counterfactual response variables as missing values, and multiply imputes them from their posterior predictive distribution, which is derived from the previously fitted regression models. Our proposed BML approach also accounts for uncertainty in the optimal decision rules at subsequent stages by integrating over their corresponding posterior distributions, which are derived from the previously fitted regression models as well. Our proposed approach uses all of the available information while accounting for the uncertainty due to the counterfactual response variables for sample observations corresponding to sub-optimal actions at subsequent stages, and the uncertainty due to the optimal decision rules at subsequent stages being unknown. By accounting for the uncertainty in the optimal decision rules, our proposed BML approach obviates the non-regular inferential challenges associated with Q-learning (see, e.g., Chakraborty et al., 2010). Because the Bayesian regression models may be quite general, with their particular forms chosen to accommodate the structure of the data set being analyzed, the proposed BML approach facilitates using Bayesian regression modeling to investigate DTRs, including robust

nonparametric modeling.

The remainder of the article is as follows. In Section 2, we use the two-stage setting to motivate the problem, establish notation, and describe limitations of many existing approaches. In Section 3, we formalize the BML approach by giving modeling requirements and defining the posterior quantities that are used to identify optimal DTRs and measure uncertainty. In Section 4, we describe the backward induction Gibbs (BIG) sampler for actually applying the BML approach in practice. In Section 6, we report the results of two simulation studies that contrast various implementations of the BML approach and comparable versions of Q-learning. In Section 7, we conclude with a brief discussion.

2 Background and Notation

To focus on the main ideas, we restrict attention to the two-stage setting. Extending the proposed approach to the general K-stage setting is formally straightforward, but requires additional regression modeling and computation. Let $A_k \in \mathcal{A}_k$ denote the action taken at stage k, for $k = 1, 2$, $O_1 \in \mathcal{O}_1$ denote baseline covariates, observed before stage 1, $O_2 \in \mathcal{O}_2$ interim covariates, observed or updated between stages 1 and 2, and $Y \in \mathcal{Y}$ the *payoff*. Depending on the particular application, part of the payoff Y may be observed between stages 1 and 2. In this case, we let $Y = Y_1 + Y_2$, where $Y_1 \in \mathcal{Y}_1$ is observed between stages 1 and 2, and $Y_2 \in \mathcal{Y}_2$ is observed after stage 2. For example, if the payoff Y is overall survival and stage 2 initiates at disease progression, then Y_1 is the time to disease progression, and Y_2 is the time from disease progression to death, so that $Y = Y_1 + Y_2$ is overall survival. We refer to Y_1 as the initial payoff, and Y_2 as the subsequent payoff. In other applications, no part of Y is observed between stages 1 and 2; rather, Y is observed only after stage 2. In this case, we let $Y_1 = 0$ and thus $Y = Y_2$. For example, in a study of two-stage anti-hypertensive drug regimes, Y may be systolic blood pressure at the end of stage 2. In the two-stage setting, a

general schematic for the full sequence of possible observables and actions is

$$O_1 \to A_1 \to (Y_1, O_2) \to A_2 \to Y_2.$$

We use an over-bar to denote accumulated history, with $\overline{O}_2 = (O_1, A_1, Y_1, O_2) \in \overline{\mathcal{O}}_2$ the patient's history at stage 2, prior to A_2, and $\overline{O}_1 = O_1$ the history at the start of stage 1, prior to A_1. Because $\overline{O}_k$ is observed before A_k, $k = 1, 2$, it can be used to choose A_k. There are many multi-stage settings, and we envision that the BML approach can be used for investigating DTRs in any such setting.

The multi-stage inferential problem is defined in terms of the following parameters. Given O_1, we denote the mean initial payoff under action a_1 at stage 1 by

$$\mu_1(a_1 \,|\, O_1) = E[Y_1 \,|\, O_1, A_1 = a_1].$$

Given $\overline{O}_2$, we denote the mean subsequent payoff under action a_2 at stage 2 by

$$\mu_2(a_2 \,|\, \overline{O}_2) = E[Y_2 \,|\, \overline{O}_2, A_2 = a_2].$$

Given O_1, we denote the mean payoff under action a_1 at stage 1 and a_2 at stage 2 by

$$\begin{aligned}
\mu(a_1, a_2 \,|\, O_1) &= E[Y \,|\, O_1, A_1 = a_1, A_2 = a_2] \\
&= E[Y_1 \,|\, O_1, A_1 = a_1] + E[Y_2 \,|\, O_1, A_1 = a_1, A_2 = a_2] \\
&= \mu_1(a_1 \,|\, O_1) + \int \mu_2(a_2 \,|\, O_1, A_1 = a_1, Y_1 = y_1, O_2 = o_2)\, g_2(y_1, o_2 \,|\, O_1, A_1 = a_1)\, d(y_1, o_2),
\end{aligned}$$

where we denote the joint conditional distribution of (Y_1, O_2) given (O_1, A_1) as G_2. In the sequel, without loss of generality we assume that larger payoffs are better. Therefore, if $\mu_2(a_2 \,|\, \overline{O}_2) > \mu_2(a_2' \,|\, \overline{O}_2)$, then a_2 is a better action than a_2' for a patient with current history $\overline{O}_2$ at stage 2, and similarly for other comparisons.

In the two-stage setting, a DTR consists of two decision rules,

$$d_1 : \mathcal{O}_1 \to \mathcal{A}_1 \text{ at stage 1} \quad \text{and} \quad d_2 : \overline{\mathcal{O}}_2 \to \mathcal{A}_2 \text{ at stage 2.}$$

The decision rules, $d_k(\overline{O}_k)$, $k = 1, 2$, define a mapping from current patient history to an action at the corresponding stage. We use the notational convention of replacing a_1, a_2, or (a_1, a_2) with d_1, d_2, or $\boldsymbol{d} = (d_1, d_2)$, respectively, to mean that the action or actions are taken according to the corresponding decision rules. For example, we use $\mu_2(d_2 \,|\, \overline{O}_2)$ to denote the mean subsequent payoff when the stage 2 action is taken according to the decision rule d_2 given $\overline{O}_2$. The primary objective is to identify the optimal DTR, $\boldsymbol{d}^{opt}$, which is defined as the set of decision rules, $d_k^{opt}(\overline{O}_k)$, $k = 1, 2$, that provide the largest mean payoff. Following dynamic programming (Bellman, 1957),

$$d_2^{opt}(\overline{O}_2) = \arg \sup_{a_2 \in \mathcal{A}_2} \mu_2(a_2 \,|\, \overline{O}_2) \;\; \forall \;\; \overline{O}_2 \in \overline{\mathcal{O}}_2,$$

$$d_1^{opt}(O_1) = \arg \sup_{a_1 \in \mathcal{A}_1} \mu(a_1, d_2^{opt} \,|\, O_1) \;\; \forall \;\; O_1 \in \mathcal{O}_1.$$

That is, $d_2^{opt}(\overline{O}_2)$ assigns the stage 2 action that achieves the largest mean subsequent payoff, Y_2, given $\overline{O}_2$, and $d_1^{opt}(O_1)$ assigns the stage 1 action that achieves the largest mean payoff, $Y = Y_1 + Y_2$, given O_1 and assuming that the optimal action is taken at stage 2. In particular, d_1^{opt} depends on d_2^{opt}, but not conversely. Because the above quantities are unknown in practice, they must be estimated using observed data.

Many approaches have been proposed for identifying optimal DTRs. Q-learning (Watkins, 1989) is a parametric approach for optimizing DTRs that is easy to implement and understand, and that has received a great deal of attention (Murphy, 2005). In the two-stage setting, Q-learning requires regression models for the two Q-functions, which are related to the previously defined quantities as follows,

$$Q_2(\overline{O}_2, A_2) = \mu_2(A_2 \,|\, \overline{O}_2),$$

$$Q_1(O_1, A_1) = \mu(A_1, d_2^{opt} \,|\, O_1) = \mu_1(A_1 \,|\, O_1) + \sup_{a_2 \in \mathcal{A}_2} \mu_2(a_2 \,|\, \overline{O}_2).$$

In practice, approximate dynamic programming is used to estimate the optimal DTR (see, e.g., Nahum-Shani et al., 2012). Because linear regression models typically are used for the Q-functions, model misspecification is a concern.

Murphy (2003) and Robins (2004) proposed semiparametric approaches that are more robust to model misspecification than Q-learning. Moodie et al. (2007) provides a review of these approaches, and shows that they are closely related. Briefly, both approaches iteratively identify optimal decision rules by estimating contrasts like $\mu_2(a_2 \,|\, \overline{O}_2) - \mu_2(d_2^{ref} \,|\, \overline{O}_2)$ and $\mu(a_1, d_2^{opt} \,|\, O_1) - \mu(d_1^{ref}, d_2^{opt} \,|\, O_1)$, where d_k^{ref}, $k = 1, 2$ are particular reference decision rules. These contrasts often are referred to as "blip" functions (Robins, 2004) or "regrets" (Murphy, 2003). As Chakraborty et al. (2010) discuss, Q-learning and these semi-parametric approaches suffer from non-regularity due to the need to compute the supremum over $\mathcal{A}_2$ in the definitions of $Q_1(O_1, A_1)$ and $\mu(a_1, d_2^{opt} \,|\, O_1) - \mu(d_1^{ref}, d_2^{opt} \,|\, O_1)$ in the context of frequentist inference. As a result, these methods encounter difficulties in quantifying uncertainty about $Q_1(O_1, A_1)$ and $\mu(a_1, d_2^{opt} \,|\, O_1) - \mu(d_1^{ref}, d_2^{opt} \,|\, O_1)$.

Zhao et al. (2015) proposed three nonparametric methods for optimizing DTRs based on outcome weighted learning (OWL), which treats identifying optimal decision rules as a classification problem. In contrast with the above methods, which infer optimal decision rules from certain models fit to the observed data, these OWL methods retrospectively investigate the differences between subjects with observed high and low rewards to determine what the optimal actions should be relative to the actual actions taken for different groups of patients (Zhao et al., 2015). In the two-stage setting, the OWL methods derive the optimal DTR by maximizing the mean payoff through the identity

$$\mu(\boldsymbol{d}) = E\left[\frac{\left(\sum_{k=1}^{2} Y_k\right)\prod_{k=1}^{2} 1\{A_k = d_k(\overline{O}_k)\}}{\prod_{k=1}^{2} \text{Prob}(A_k = a_k \,|\, \overline{O}_k)}\right],$$

where $1\{\cdot\}$ is a binary indicator and $\mu(\boldsymbol{d})$ denotes the mean payoff under $\boldsymbol{d}$. An important limitation shared by the backward (BOWL) and iterated (IOWL) versions of the OWL approach is that, to optimize the stage 1 action, these methods only use the sample observations corresponding to the estimated optimal stage 2 action. Because the number of sample observations corresponding to the optimal actions for a particular set of decision rules quickly diminishes as the number of stages increase, BOWL and IOWL can be inefficient.

OWL methods, and the related doubly-robust augmented value search method of Zhang et al. (2013), also are limited by the fact that they do not easily accommodate more than two actions at each stage.

Many likelihood-based methods, both Bayesian and frequentist, have been used in practice to identify optimal DTRs. See, for example, Thall et al. (2000), Thall et al. (2007), Arjas and Saarela (2010), Zajonc (2012), Lee et al. (2015), and Xu et al. (2016). Likelihood-based methods estimate the joint distribution of (Y_2, O_2, Y_1) given (A_2, A_1, O_1), and then either apply dynamic programming or a full numerical search of the action space to identify the optimal DTR, which can be non-trivial. Typically, a sequence of conditional regression models are fitted to the observed data, e.g., Y_2 given $(O_1, A_1, Y_1, O_2, A_2)$, O_2 given (O_1, A_1, Y_1), and Y_1 given (O_1, A_1). Chakraborty and Murphy (2014) refer to this as a "dynamical systems model." Likelihood-based methods require modeling that becomes increasingly complex in the dimension of O_2, which can be large in practice. This problem becomes worse as the number of stages increases as well. Moreover, the conditional distribution of O_2 given (O_1, A_1, Y_1) is not of direct interest, and misspecifying this aspect of the model may prove detrimental for identifying the optimal DTR. The greater modeling burden of likelihood-based methods is a disadvantage compared to the other approaches mentioned above.

Our proposed BML approach obviates many of the challenges associated with the approaches mentioned above, in the following ways:

(i) In contrast with Q-learning and the related semiparametric approaches, it produces measures of uncertainty with good coverage in non-regular settings.

(ii) In contrast with value search methods like OWL, it uses the available information from sample observations that did not follow the DTR under investigation.

(iii) In contrast with likelihood-based methods, it does not require directly modeling the covariate processes.

Our proposed BML approach is most closely related to approximate dynamic programming

methods like Q-learning, but it naturally exploits flexible Bayesian regression modeling. This helps to alleviate concerns about model misspecification, a major criticism of Q-learning. We envision that our proposed BML approach can be used to develop methods for identifying optimal DTRs in many settings with various payoff structures, including real-valued, count, binary, and time-to-event outcomes. We focus on settings with payoffs that are either real-valued or binary, however.

3 Proposed Approach

To develop the proposed BML approach, we use the potential outcome notation of Rubin (1974). We let $Y(a_1, a_2)$ denote the payoff that would be observed when action a_1 is taken at stage 1 and action a_2 is taken at stage 2. We similarly denote the three other potential outcomes in the two-stage setting, $Y_2(a_1, a_2)$, $Y_1(a_1)$, and $O_2(a_1)$. We assume the potential outcomes are consistent, i.e., an observed outcome corresponds to the potential outcome for the action(s) actually taken. For example, if we observe $(o_1, a_1, y_1, o_2, a_2, y_2)$, then $Y_1(a_1) = y_1$, $O_2(a_1) = o_2$, $Y_2(a_1, a_2) = y_2$, and $Y(a_1, a_2) = y$ $(= y_1 + y_2)$. We assume $Prob(A_k = a_k \,|\, \overline{O}_k)$, $k = 1, 2$, does not depend on subsequent potential outcomes, i.e., the sequential randomization assumption. These assumptions facilitate identifying the causal effects of the actions at each stage (Rubin, 1990).

We use $Y\left(a_1, d_2^{opt}\right)$ to denote the payoff that would be observed when action a_1 is taken at stage 1 and the corresponding optimal action is taken at stage 2 as stipulated by d_2^{opt}, which depends on the patient history at stage 2 that would be observed when action a_1 is taken at stage 1, i.e., $\overline{O}_2(a_1) = (o_1, a_1, Y_1(a_1), O_2(a_1))$. We define $Y_2\left(a_1, d_2^{opt}\right)$ similarly. We use $Y_1(d_1^{opt})$ and $O_2(d_1^{opt})$ to denote the interim payoff and covariates that would be observed at stage 2 when the optimal action is taken at stage 1 as stipulated by d_1^{opt}. We use $Y\left(\boldsymbol{d}^{opt}\right)$ to denote the payoff that would be observed when the optimal actions are taken at both stages 1 and 2. We define $Y_2\left(\boldsymbol{d}^{opt}\right)$ similarly.

A brief overview of the proposed approach is as follows. The stage 2 regression model uses $Y_2(A_1, A_2)$ as the response variable and $(\overline{O}_2, A_2)$ as covariates. The stage 1 regression model uses $Y\left(A_1, d_2^{opt}\right)$ as the response variable and (O_1, A_1) as covariates. The stage 2 regression model is estimated first, and it is used to identify d_2^{opt} and the relevant potential payoff that defines the response variable in the stage 1 regression model for each sample observation. For example, if $d_2^{opt}(\overline{o}_2) = a_2$, then $Y(a_1, a_2)$ is the relevant stage 1 response variable for a sample observation with $\overline{O}_2 = \overline{o}_2 = (o_1, a_1, y_1, o_2)$. The stage 1 regression model is estimated second, and it is used to identify d_1^{opt}. Together, the fitted stage 1 and 2 regression models provide a statistical basis for identifying the optimal two-stage DTR, and computing other quantities of interest, including the posterior probability that a certain action is optimal for a specific patient history at a particular stage.

As described in detail below, for a sample observation with $(o_1, a_1, y_1, o_2, a_2, y_2)$, the consistency assumption ensures that $\overline{O}_2(a_1) = \overline{o}_2$. Therefore, given d_2^{opt}, the relevant potential payoff that defines the stage 1 response variable, i.e., $Y\left(a_1, d_2^{opt}\right)$, can be determined for this observation. This critical result facilitates actually fitting the stage 1 regression model to a particular set of sample observations. The value of $Y\left(a_1, d_2^{opt}\right)$ may or may not be observed, however. In particular, for a sample observation corresponding to the optimal action at stage 2, $Y\left(a_1, d_2^{opt}\right)$ is the observed payoff. By contrast, for a sample observation corresponding to a sub-optimal action at stage 2, $Y\left(a_1, d_2^{opt}\right)$ is counterfactual. Because the posterior predictive distribution for this unobserved payoff can be derived from the previously fitted stage 2 regression model, we average over this distribution when fitting the stage 1 regression model. Moreover, because the posterior distribution for d_2^{opt} can be derived from the previously fitted stage 2 regression model, we average over this distribution when fitting the stage 1 regression model as well. Thus, the stage 2 regression model has a dual purpose, to provide information about (i) the optimal decision rule at stage 2, and (ii) the counterfactual response variables in the stage 1 regression model that result for a particular decision rule at stage 2. The proposed approach accounts for uncertainty in the optimal

stage 2 decision rule and incorporates all the available information about the value of the counterfactual response variables when fitting the stage 1 regression model. We formalize the proposed approach below, and then in Section 4 we describe an efficient sampling algorithm that facilitates using the proposed approach in practice.

3.1 Stage 2 Regression Model

Our BML approach requires a stage 2 regression model for the distribution of the subsequent payoff conditional on the current history and the viable actions at stage 2. This model is defined generally as follows,

$$\begin{aligned} Y_2(A_1, A_2) \,|\, \overline{O}_2, A_2; \boldsymbol{\theta}_2 &\sim f_2(y_2 \,|\, \overline{O}_2, A_2; \, \boldsymbol{\theta}_2), \\ \boldsymbol{\theta}_2 &\sim p_{2,0}(\boldsymbol{\theta}_2), \end{aligned} \tag{1}$$

where $\boldsymbol{\theta}_2$ is an unknown parameter vector, with prior distribution $p_{2,0}$. This model can be a standard Bayesian regression model based on a linear predictor although, if desired, a Bayesian nonparametric regression model can be used to enhance robustness (see, e.g., Müller et al., 2015). We discuss choosing a specific f_2 and $p_{2,0}$ in greater detail below, in Section 5. Denote the observed data as $\boldsymbol{D}_n = \{(o_{1,i}, a_{1,i}, y_{1,i}, o_{2,i}, a_{2,i}, y_{2,i})\}_{i=1}^n$, which we assume consists of n independent sample observations. The consistency assumption implies that $Y_{2,i}(a_{1,i}, a_{2,i}) = y_{2,i}$ and $\overline{O}_{2,i}(a_{2,i}) = \overline{o}_{2,i}$, for $i = 1, \ldots, n$. Therefore, applying Bayes theorem, the stage 2 posterior distribution is

$$p_{2,n}(\boldsymbol{\theta}_2 \,|\, \boldsymbol{D}_n) \propto p_{2,0}(\boldsymbol{\theta}_2) \times \prod_{i=1}^{n} f_2(y_{2,i} \,|\, \overline{O}_2 = \overline{o}_{2,i}, A_2 = a_{2,i}; \, \boldsymbol{\theta}_2), \tag{2}$$

and the stage 2 posterior predictive distribution is

$$f_{2,n}(y_2 \,|\, \overline{O}_2, A_2; \, \boldsymbol{D}_n) = \int f_2(y_2 \,|\, \overline{O}_2, A_2; \, \boldsymbol{\theta}_2) \, p_{2,n}(\boldsymbol{\theta}_2 \,|\, \boldsymbol{D}_n) \, d\boldsymbol{\theta}_2. \tag{3}$$

Other posterior quantities arising from the stage 2 model that we use later are as follows.

Given $\boldsymbol{\theta}_2$, the model-based optimal decision rule at stage 2 is

$$d_2^{opt}(\overline{O}_2;\boldsymbol{\theta}_2) = \arg\sup_{a_2\in\mathcal{A}_2} \mu_2(a_2\,|\,\overline{O}_2;\boldsymbol{\theta}_2) = \arg\sup_{a_2\in\mathcal{A}_2} E[Y_2(A_1,A_2)\,|\,\overline{O}_2, A_2=a_2;\,\boldsymbol{\theta}_2],$$

where the expectation is with respect to $f_2(y_2\,|\,\overline{O}_2, A_2;\,\boldsymbol{\theta}_2)$ defined in (1), i.e., the conditional density function assumed for the subsequent payoff in the stage 2 regression model. Because $d_2^{opt}(\overline{O}_2;\boldsymbol{\theta}_2)$ depends on $\boldsymbol{\theta}_2$, the optimal decision rule at stage 2 has a posterior distribution, which we denote as $p_{2,n}(d_2^{opt}\,|\,\boldsymbol{D}_n)$. The posterior optimal decision rule at stage 2 is

$$\widehat{d}_2^{opt}(\overline{O}_2) = \arg\sup_{a_2\in\mathcal{A}_2} \widehat{\mu}_2(a_2\,|\,\overline{O}_2) = \arg\sup_{a_2\in\mathcal{A}_2} E[Y_2(A_1,A_2)\,|\,\overline{O}_2, A_2=a_2;\,\boldsymbol{D}_n], \tag{4}$$

where the expectation is taken with respect to $f_{2,n}(y_2\,|\,\overline{O}_2, A_2;\,\boldsymbol{D}_n)$ defined in (3), i.e., the stage 2 posterior predictive distribution. In (4), $\widehat{\mu}_2(a_2\,|\,\overline{O}_2)$ is the posterior analog of the previously defined $\mu_2(a_2\,|\,\overline{O}_2)$, i.e., the mean subsequent payoff when $A_2 = a_2$ given $\overline{O}_2$. Given $\overline{O}_2$, the posterior probability that a_2 is the optimal action at stage 2 is

$$\begin{aligned}\pi(a_2\,|\,\overline{O}_2;\boldsymbol{D}_n) &= \text{Prob}\left[a_2 = d_2^{opt}(\overline{O}_2)\,|\,\overline{O}_2;\boldsymbol{D}_n\right] \\ &= \int 1\left\{a_2 = d_2^{opt}(\overline{O}_2;\boldsymbol{\theta}_2)\right\}\, p_{2,n}(\boldsymbol{\theta}_2\,|\,\boldsymbol{D}_n)\,d\boldsymbol{\theta}_2.\end{aligned} \tag{5}$$

We refer to $\pi(a_2\,|\,\overline{O}_2;\boldsymbol{D}_n)$ as the "posterior optimality probability" of action a_2 at stage 2 for a patient with $\overline{O}_2$.

3.2 Stage 1 Regression Model

Our BML approach requires a stage 1 regression model for the distribution of the payoff conditional on the baseline covariates and viable actions at stage 1, assuming that the optimal action will be taken at stage 2. This model is defined generally as follows,

$$\begin{aligned} Y\left(A_1, d_2^{opt}\right)\,|\,O_1, A_1;\boldsymbol{\theta}_1 &\overset{ind}{\sim} f_1(y\,|\,O_1, A_1;\,\boldsymbol{\theta}_1), \\ \boldsymbol{\theta}_1 &\sim p_{1,0}(\boldsymbol{\theta}_1), \end{aligned} \tag{6}$$

where $\boldsymbol{\theta}_1$ is an unknown parameter vector, with prior $p_{1,0}$. Because the stage 1 regression model does not include the interim variables $(Y_1(A_1), O_2(A_1))$ as covariates, it facilitates causal inference for the stage 1 actions, assuming the corresponding optimal action will be taken at stage 2. However, the stage 1 response variable, $Y\left(A_1, d_2^{opt}\right)$, depends on $(Y_1(A_1), O_2(A_1))$ through d_2^{opt}. Therefore, in order to actually fit this model using the observed data, we must establish that it is possible to identify the relevant stage 1 response variable corresponding to each sample observation.

To explain our approach, we assume temporarily that d_2^{opt} is known. Once we have established the proposed BML approach for the case where d_2^{opt} is known, we will explain how to deal with the case where d_2^{opt} is not known. We rely on the assumption that the potential outcomes are consistent, which implies that $Y_{1,i}(a_{1,i}) = y_{1,i}$ and $O_{2,i}(a_{1,i}) = o_{2,i}$, for $i = 1, \ldots, n$. Therefore, $a_{2,i}^{opt} = d_2^{opt}(\overline{o}_{2,i})$ is the optimal action at stage 2 corresponding to the ith sample observation, and thus, $\{Y_i\left(a_{1,i}, a_{2,i}^{opt}\right)\}_{i=1}^n$ are the relevant stage 1 response variables corresponding to the observed data. The value of each $Y_i\left(a_{1,i}, a_{2,i}^{opt}\right)$, $i = 1, \ldots, n$, may or may not be known, however. In particular, if $a_{2,i}^{opt} = a_{2,i}$, then $Y_i\left(a_{1,i}, a_{2,i}^{opt}\right) = y_i$. By contrast, if $a_{2,i}^{opt} \neq a_{2,i}$, then $Y_i\left(a_{1,i}, a_{2,i}^{opt}\right) = y_{1,i} + Y_{2,i}\left(a_{1,i}, a_{2,i}^{opt}\right)$, where $Y_{2,i}\left(a_{1,i}, a_{2,i}^{opt}\right)$ is counterfactual. Therefore, if a sample observation corresponds to the optimal action at stage 2, then the value of the stage 1 response variable for this sample observation is known; whereas, if a sample observation corresponds to a sub-optimal action at stage 2, then the value of the stage 1 response variable for this sample observation is not known. The key insight is that we can use the previously fitted stage 2 regression model to derive the posterior predictive distribution for the counterfactual stage 1 response variables, $\{Y_{2,i}\left(a_{1,i}, a_{2,i}^{opt}\right) : a_{2,i} \neq a_{2,i}^{opt}\}_{i=1}^n$. Our proposed BML approach exploits the stage 2 regression model when calculating the stage 1 posterior distribution by averaging over the posterior predictive distribution for the counterfactual stage 1 response variables.

To ease notation, we denote $\boldsymbol{y}_2^{mis} = \{Y_{2,i}\left(a_{1,i}, a_{2,i}^{opt}\right) : a_{2,i} \neq a_{2,i}^{opt}\}_{i=1}^n$. The stage 1 posterior

distribution is

$$p_{1,n}\left(\boldsymbol{\theta}_1 \,|\, d_2^{opt};\, \boldsymbol{D}_n\right) \propto p_{1,0}(\boldsymbol{\theta}_1) \quad \times \prod_{\{i:a_{2,i}=a_{2,i}^{opt}\}} f_1\left(y_{1,i}+y_{2,i} \,|\, o_{1,i}, a_{1,i};\, \boldsymbol{\theta}_1\right)$$
$$\times \int \left[\prod_{\{i:a_{2,i}\neq a_{2,i}^{opt}\}} f_1\left(y_{1,i}+y_{2,i}^{mis} \,|\, o_{1,i}, a_{1,i}; \boldsymbol{\theta}_1\right) \right] f_{2,n}\left(\boldsymbol{y}_2^{mis} \,|\, \boldsymbol{D}_n\right)\, d\boldsymbol{y}_2^{mis}, \tag{7}$$

where $f_1(y \,|\, O_1, A_1; \boldsymbol{\theta}_1)$ and $p_{1,0}(\boldsymbol{\theta}_1)$ are specified in (6), and $f_{2,n}(\boldsymbol{y}_2^{mis} \,|\, \boldsymbol{D}_n)$ is the posterior predictive distribution for the counterfactual stage 1 response variables, $\boldsymbol{y}_2^{mis}$. The stage 1 posterior predictive distribution is

$$f_{1,n}\left(y \,|\, O_1, A_1, d_2^{opt};\, \boldsymbol{D}_n\right) = \int f_1\left(y \,|\, O_1, A_1; \boldsymbol{\theta}_1\right)\, p_{1,n}\left(\boldsymbol{\theta}_1 \,|\, d_2^{opt}; \boldsymbol{D}_n\right)\, d\boldsymbol{\theta}_1. \tag{8}$$

Because the stage 1 posterior distribution in (7), and the stage 1 posterior predictive distribution in (8), depends on the optimal stage 2 decision rule through the stage 1 response variable, we include d_2^{opt} to the right of the conditioning bar.

Given $\boldsymbol{\theta}_1$, the model-based optimal decision rule at stage 1 is

$$d_1^{opt}(O_1; \boldsymbol{\theta}_1) = \arg \sup_{a_1 \in \mathcal{A}_1} \mu(a_1, d_2^{opt} \,|\, O_1; \boldsymbol{\theta}_1) = \arg \sup_{a_1 \in \mathcal{A}_1} E[Y(A_1, d_2^{opt}) \,|\, O_1, A_1 = a_1;\, \boldsymbol{\theta}_1],$$

where the expectation is with respect to $f_1(y \,|\, O_1, A_1;\, \boldsymbol{\theta}_1)$ defined in (6), i.e., the conditional density function assumed for the payoff in the stage 1 regression model. Because $d_1^{opt}(O_1; \boldsymbol{\theta}_1)$ depends on $\boldsymbol{\theta}_1$, the optimal decision rule at stage 1 has a posterior distribution, which we denote as $p_{1,n}(d_1^{opt} \,|\, \boldsymbol{D}_n)$. The posterior optimal decision rule at stage 1 is

$$\widehat{d}_1^{opt}(O_1) = \arg \sup_{a_1 \in \mathcal{A}_1} \widehat{\mu}\left(a_1, d_2^{opt} \,|\, O_1\right) = \arg \sup_{a_1 \in \mathcal{A}_1} E\left[Y\left(A_1, d_2^{opt}\right) \,\middle|\, O_1, A_1 = a_1;\, \boldsymbol{D}_n\right], \tag{9}$$

where the expectation is taken with respect to $f_{1,n}(y \,|\, O_1, A_1, d_2^{opt}; \boldsymbol{D}_n)$ defined in (7), i.e., the stage 1 posterior predictive distribution. In (9), $\widehat{\mu}\left(a_1, d_2^{opt} \,|\, O_1\right)$ is the posterior analog of the previously defined $\mu(a_1, d_2^{opt} \,|\, O_1)$, i.e., the mean payoff when $A_1 = a_1$ and the optimal action is taken at stage 2, given O_1. While $\widehat{d}_1^{opt}(O_1)$ in (9) is a mapping from the baseline covariates to a stage 1 action, it explicitly reflects the assumption that the optimal action

is taken at stage 2. The posterior optimal two-stage DTR, $\widehat{\boldsymbol{d}}^{opt}$, consists of the posterior optimal rules at stages 1 and 2, i.e., $\widehat{d}_k^{opt}(\overline{O}_k)$, $k = 1, 2$. The posterior optimality probability of action a_1 at stage 1 for a patient with O_1 is

$$\begin{aligned}\pi(a_1 \,|\, O_1; \boldsymbol{D}_n) &= \text{Prob}\left[a_1 = d_1^{opt}(O_1) \,\middle|\, d_2^{opt}; \boldsymbol{D}_n\right] \\ &= \int 1\left\{a_1 = d_1^{opt}(O_1; \boldsymbol{\theta}_1)\right\}\, p_{1,n}(\boldsymbol{\theta}_1 \,|\, d_2^{opt}; \boldsymbol{D}_n)\, d\boldsymbol{\theta}_1.\end{aligned} \tag{10}$$

So far, the derivations of the stage 1 posterior quantities have relied on the assumption that the optimal decision rule at stage 2 is known. To deal with the case where the optimal decision rule at stage 2 is not known, we calculate the stage 1 posterior distribution using the model-based optimal stage 2 decision rule, $d_2^{opt}(\overline{O}_2 \,|\, \boldsymbol{\theta}_2)$, and average over the stage 2 posterior distribution for $\boldsymbol{\theta}_2$. That is, when d_2^{opt} is unknown, the stage 1 posterior distribution is

$$\begin{aligned}p_{1,n}\left(\boldsymbol{\theta}_1 \,|\, d_2^{opt};\, \boldsymbol{D}_n\right) \propto p_{1,0}(\boldsymbol{\theta}_1) \quad &\times \int \left\{ \prod_{\{i:a_{2,i}=d_2^{opt}(\overline{o}_{2,i};\boldsymbol{\theta}_2)\}} f_1(y_{1,i} + y_{2,i} \,|\, o_{1,i}, a_{1,i};\, \boldsymbol{\theta}_1) \right. \\ \times \int \left[\prod_{\{i:a_{2,i}\neq d_2^{opt}(\overline{o}_{2,i};\boldsymbol{\theta}_2)\}} f_1(y_{1,i} + y_{2,i}^{mis} \,|\, o_{1,i}, a_{1,i}; \boldsymbol{\theta}_1) \right] &\left. f_{2,n}(\boldsymbol{y}_2^{mis} \,|\, \boldsymbol{D}_n)\, d\boldsymbol{y}_2^{mis} \right\} p_{2,n}(\boldsymbol{\theta}_2 \,|\, \boldsymbol{D}_n) d\boldsymbol{\theta}_2,\end{aligned} \tag{11}$$

where $p_{2,n}(\boldsymbol{\theta}_2 \,|\, \boldsymbol{D}_n)$ is the stage 2 posterior distribution defined in (2). The other stage 1 posterior distributions and quantities retain the same definitions as before, but are derived with respect to the revised stage 1 posterior distribution in (11). In this way, the stage 1 posterior distribution accounts for uncertainty that results from the optimal decision rule at stage 2 being unknown.

4 The BIG Sampler for Posterior Estimation

Because the response variables in the stage 2 regression model are not counterfactual, sampling from and estimating the stage 2 posterior distribution defined in (2) is feasible using standard Bayesian methods and software (see, e.g., Carlin and Louis, 2009). By contrast, because the response variables in the stage 1 regression model may be counterfactual, sampling

from and estimating the stage 1 posterior distribution defined in (11) is not immediately feasible using standard Bayesian methods and software. To address this problem, we propose a general sampling algorithm, which we call the backward induction Gibbs (BIG) sampler. This sampling algorithm greatly facilitates computation through Bayesian data augmentation. In particular, we sample an optimal stage 2 decision rule from its posterior distribution, and then we sample the counterfacual response variables in the stage 1 regression model that result for this rule from their joint posterior predictive distribution. Augmenting the data set in this way makes sampling from the stage 1 posterior distribution feasible using standard methods, and thus the BIG sampler is a practical tool for actually using the proposed BML approach in practice.

The BIG sampler consists of three steps:

Step 1. Sample $\boldsymbol{\theta}_2 \sim p_{2,n}(\boldsymbol{\theta}_2 \,|\, \boldsymbol{D}_n)$ and set $a_{2,i}^{opt} = d_2^{opt}(\overline{o}_{2,i}; \boldsymbol{\theta}_2)$, for $i = 1, \ldots, n$, where $p_{2,n}(\theta_2 \,|\, \boldsymbol{D}_n)$ is the stage 2 posterior distribution and $d_2^{opt}(\overline{o}_{2,i}; \boldsymbol{\theta}_2)$ is the model-based optimal decision rule at stage 2.

Step 2. Sample $\boldsymbol{y}_2^{mis} \sim f_{2,n}\left(\boldsymbol{y}_2^{mis} \,|\, \boldsymbol{D}_n\right)$, where $\boldsymbol{y}_2^{mis} = \{Y_{2,i}(a_{1,i}, a_{2,i}^{opt}) : a_{2,i} \neq a_{2,i}^{opt}\}_{i=1}^n$, and $f_{2,n}(\boldsymbol{y}_2^{mis} \,|\, \boldsymbol{D}_n)$ is the posterior predictive distribution for $\boldsymbol{y}_2^{mis}$. For $i = 1, \ldots, n$, if $a_{2,i} = a_{2,i}^{opt}$, then set $y_{2,i}^{opt} = y_{2,i}$, and if $a_{2,i} \neq a_{2,i}^{opt}$, then set $y_{2,i}^{opt} = y_{2,i}^{mis}$.

Step 3. Sample $\boldsymbol{\theta}_1 \sim p_{1,n}\left(\boldsymbol{\theta}_1 \,|\, \boldsymbol{D}_n^{opt}\right)$, where $\boldsymbol{D}_n^{opt} = \{(o_{1,i}, a_{1,i}, y_{1,i}, y_{2,i}^{opt}\}_{i=1}^n$,

$$p_{1,n}\left(\boldsymbol{\theta}_1 \,|\, \boldsymbol{D}_n^{opt}\right) \propto p_{1,0}\left(\boldsymbol{\theta}_1\right) \times \prod_{i=1}^{n} f_1\left(y_{1,i} + y_{2,i}^{opt} \,|\, o_{1,i}, a_{1,i}; \boldsymbol{\theta}_1\right),$$

$f_1\left(y \,|\, O_1, A_1; \boldsymbol{\theta}_1\right)$ is the conditional density function assumed for the payoff in the stage 1 regression model, and $p_{1,0}\left(\boldsymbol{\theta}_1\right)$ is the prior for $\boldsymbol{\theta}_1$ in the stage 1 regression model.

In Step 1, $\boldsymbol{\theta}_2$ is sampled from its posterior distribution, and the optimal stage 2 action for each sample observation is determined using the corresponding model-based optimal decision rule at stage 2. In Step 2, given the optimal actions at stage 2 that were determined

in Step 1, the counterfactual response variables in the stage 1 regression model are sampled from their posterior predictive distribution, which is derived from the previously fitted stage 2 regression model as well. Depending on the specific stage 2 regression model, often this step can be carried out first by sampling $\boldsymbol{\theta}_2 \sim p_{2,n}(\boldsymbol{\theta}_2 \,|\, \boldsymbol{D}_n)$, and second by sampling $y_{2,i}^{mis} \sim f_{2,n}(y_{2,i}^{mis} \,|\, \boldsymbol{\theta}_2; \boldsymbol{D}_n)$, for $i = 1, \ldots, n$. In Step 3, $\boldsymbol{\theta}_1$ is sampled from its full posterior distribution conditional on the augmented data set $\boldsymbol{D}_n^{opt}$ that was obtained in Steps 1 and 2. Critically, the full conditional stage 1 posterior distribution, $p_{1,n}\left(\boldsymbol{\theta}_1 \,|\, \boldsymbol{D}_n^{opt}\right)$, has the same structure as the posterior distribution that would arise when the response variables in the stage 1 regression model are not counterfactual. Therefore, sampling from $p_{1,n}\left(\boldsymbol{\theta}_1 \,|\, \boldsymbol{D}_n^{opt}\right)$ is straightforward using standard Bayesian methods and software. Sampling from the stage 1 posterior distribution defined in (11) is carried out simply by iterating Steps 1-3.

Because the augmented data set, $\boldsymbol{D}_n^{opt}$, changes with each iteration of the BIG sampler, if a Markov chain Monte Carlo (MCMC) algorithm is needed in Step 3 to sample from the full conditional stage 1 posterior distribution, then posterior convergence must be attained at every iteration of the BIG sampler to ensure that the resulting samples are actually from the stage 1 posterior distribution. In practice, because $\boldsymbol{D}_n^{opt}$ does not change substantially between iterations, by initializing the MCMC algorithm in Step 3 at its previous state, posterior convergence tends to be very fast. Moreover, because information need not be shared between iterations of the BIG sampler, parallelization can be used to reduce computing time.

For completeness, we provide explicit sample-based calculations for the posterior quantities derived in Section 3. We use $\{\boldsymbol{\theta}_2^{(g)}\}_{g=1}^{G}$ to denote G samples from (2), obtained using a Gibbs sampler, and $\{\boldsymbol{\theta}_1^{(g)}\}_{g=1}^{G}$ to denote G samples from (11), obtained using the BIG sampler. To calculate the posterior quantities required by the proposed method, these samples

can be used as follows:

$$\widehat{\mu}_2(a_2 \,|\, \overline{O}_2) = G^{-1} \sum_{g=1}^{G} \mu_2 \left(a_2 \,\middle|\, \overline{O}_2;\, \boldsymbol{\theta}_2^{(g)} \right),$$

$$\pi(a_2 \,|\, \overline{O}_2; \boldsymbol{D}_n) = G^{-1} \sum_{g=1}^{G} 1 \left\{ a_2 = d_2^{opt} \left(\overline{O}_2; \boldsymbol{\theta}_2^{(g)} \right) \right\},$$

$$\widehat{\mu} \left(a_1, d_2^{opt} \,|\, O_1 \right) = G^{-1} \sum_{g=1}^{G} \mu \left(a_1, d_2^{opt} \,\middle|\, O_1;\, \boldsymbol{\theta}_1^{(g)} \right),$$

$$\pi(a_1 \,|\, O_1; \boldsymbol{D}_n) = G^{-1} \sum_{g=1}^{G} 1 \left\{ a_1 = d_1^{opt} \left(O_1; \boldsymbol{\theta}_1^{(g)} \right) \right\},$$

where $\mu_2 \left(a_2 \,\middle|\, \overline{O}_2;\, \boldsymbol{\theta}_2 \right)$, $d_2^{opt} \left(\overline{O}_2; \boldsymbol{\theta}_2 \right)$, $\mu \left(a_1, d_2^{opt} \,\middle|\, O_1;\, \boldsymbol{\theta}_1 \right)$, $d_2^{opt} \left(\overline{O}_2; \boldsymbol{\theta}_2 \right)$, and $d_1^{opt} \left(O_1; \boldsymbol{\theta}_1 \right)$ are defined previously in Section 3.

5 Establishing Regression Models and Priors

While the BML approach for optimizing DTRs and the BIG sampler for computing posteriors both are very general, as in any Bayesian data analysis the regression models and priors must be chosen carefully. This should be done to reflect both the covariates and the range of the response variable at each stage in the data set at hand, and also to ensure a reasonable degree of robustness. That is, the problem of Bayesian regression modeling is not obviated by our proposed approach. While we can not give exhaustive guidelines for regression modeling and prior specification here, we do offer advice for specifying Bayesian regression models and priors when applying the proposed BML approach in some common multi-stage settings.

The structure of the response variable at each stage is important for determining an appropriate series of regression models to implement the BML approach. Common structures are real-valued (e.g., log mean arterial pressure), binary (e.g., resolution of infection), count (e.g., number of seizures), and time-to-event (e.g., survival). A series of regression models that is appropriate for a real-valued response variable, is not appropriate for a binary, count, or time-to-event response variable. That said, the generic Bayesian regression problem is

to determine the conditional distribution of the response variable given the covariates, with particular interest in the mean. In the two-stage setting, the stage 2 Bayesian regression problem is to determine the conditional distribution of $Y_2(A_1, A_2)$ given $(\overline{O}_2, A_2)$, and the stage 1 Bayesian regression problem is to determine the conditional distribution of $Y(A_1, d_2^{opt})$ given (O_1, A_2). Because the optimal decision rules are identified through the means of these conditional distributions, it is critical that the mean in each regression model is robust. If the mean is not robust and misspecified, then the identified decision rules may be far from optimal. Motivated by this concern, we suggest implementing our BML approach with a series of Bayesian nonparametric (BNP) regression models chosen to reflect the structure of the response variable at each stage in the data set at hand. Critically, BNP regression models facilitate robust estimation of the mean, and if desired, of the entire conditional distribution. For an overview of BNP regression, see Müller et al. (2015, Section 4). We recommend some robust BNP regression models for implementing the BML approach below.

For a multi-stage setting with a real-valued payoff, Bayesian additive regression trees (BART) (Chipman et al., 2010) offer a powerful framework for implementing the BML approach. BART assumes a nonparametric mean function, and thus can identify non-linear associations and interactions between covariates and the mean of the response variable. Because we use BART in our simulation study in Section 6, we provide some details here. Using Z to denote the response variable, and x a realization of the covariates, BART assumes

$$Z = \sum_{j=1}^{m} g(x; T_j, M_j) + \epsilon, \qquad \epsilon \sim \text{Normal}(0, \sigma^2),$$

where ϵ is the residual, $g(x; T_j, M_j)$ is a binary regression tree, and m is the number of trees. Each binary regression tree is determined by a series of splitting rules (T_j) and the set of quantities associated with each terminal node (M_j). For example, if x is a single covariate, T_j might be defined by the binary splitting rule, $x < 0$ versus $x \geq 0$, and then M_j given T_j consists of the two quantities, $\mu_{j,1}$, corresponding to $x < 0$, and $\mu_{j,2}$, corresponding to $x \geq 0$. As another example, if x includes two covariates x_1 and x_2, T_j might be defined

by the binary splitting rules, $x_1 < 0$ versus $x_1 \geq 0$, and for $x_1 < 0$, $x_2 < 0$ versus $x_2 \geq 0$, and then M_j given T_j consists of the three quantities, $\mu_{j,1}$, corresponding to $x_1 < 0$ and $x_2 < 0$, $\mu_{j,2}$, corresponding to $x_1 < 0$ and $x_2 \geq 0$, and $\mu_{j,3}$, corresponding to $x_1 \geq 0$. BART assumes the mean function is a sum of m trees, and thus the mean response at a particular x is the sum of the m quantities associated with the terminal node that corresponds to x in each of the m trees. Chipman et al. (2010) show that BART is a robust model that can capture non-linear trends and interactions between the covariates and the mean response. Although they recommend a default model with $m = 200$ trees or selecting m via cross-validation, in our experience BART models with between 20 and 50 trees often are adequately flexible and computationally efficient. Other potentially useful BNP regression models for implementing the BML approach in multi-stage settings with a real-valued payoff include Gaussian processes and fully nonparametric conditional regression, see, e.g., Müller et al. (2015, Section 4); we do not explore these options in the sequel, however.

A particular BART model is determined by $(T_1, M_1), \ldots, (T_m, M_m)$ and σ, and these require careful prior specification. Chipman et al. (2010) provide detailed guidance in this regard. Briefly, they suggest specifying the joint prior such that

$$p_0((T_1, M_1), \ldots, (T_m, M_m), \sigma) = p_0(\sigma) \prod_j p_0(M_j|T_j) p_0(T_j),$$

where $p_0(M_j|T_j)$ and $p_0(T_j)$ have the same form for each $j = 1, \ldots, m$. Following Chipman et al. (1998), they define $p_0(T_j)$ such that each variable in x has an equal probability of being selected for a splitting rule, the value where the split occurs is uniformly distributed over the eligible domain for the selected covariate, and the probability that a particular split is terminal is equal to $\alpha/(1+d)^\beta$, where $\alpha \in (0, 1)$, $\beta \in [0, \infty)$, and $d \in \{0, 1, 2, \ldots\}$ is the depth of the split (i.e., the number of preceding splits). They recommend the default values $\alpha = 0.95$ and $\beta = 2$, which encourage individual trees to be small (i.e., ≤ 5 terminal nodes), but do not prohibit large individual trees *a posteriori*. They define $p_0(M_j|T_j)$ such that each quantity associated with a particular terminal node is assigned the same prior distribution,

namely a conjugate Normal(μ_0, σ_0^2) distribution. They develop a strategy for specifying μ_0 and σ_0 based on the prior probability that $E[Z \,|\, X = x]$ is between the minimum and maximum for the response variable in the data set at hand, and they recommend setting this prior probability equal to 0.95. For $p_0(\sigma)$, they recommend using a conjugate inverse-chi-square distribution with a substantial prior probability, e.g., 0.90, that σ is less than the sample standard deviation. Posterior inference for BART can be carried out using the `R` package `BayesTree`.

For similar reasons described above, the probit BART model (Chipman et al., 2010) is appealing for implementing the BML approach in a multi-stage setting with a binary payoff. Because we use the probit BART model in one of our simulation studies in Section 6, we provide some details here. The probit BART model assumes

$$\text{Prob}\,(Z = 1 \,|\, x) = \Phi\left[\sum_{j=1}^{m} g(x; T_j, M_j)\right],$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function. The probit BART model is determined by $(T_1, M_1), \ldots, (T_m, M_m)$, and Chipman et al. (2010) suggest a similar prior as before. One difference is that they recommend specifying $p(M_j|T_j)$ based on the prior probability that $\text{Prob}\,(Z = 1 \,|\, X = x)$ is between $\Phi(-3+q)$ and $\Phi(3+q)$, where q is an offset such that the prior mean is $\Phi(q)$. Posterior inference for the probit BART model also can be carried out using the `BayesTree` package in `R`.

For multi-stage settings with a time-to-event payoff, the BML approach could be implemented with BART survival models proposed by Sparapani et al. (2016), or dependent Dirichlet process models with Gaussian process base measures (DDP-GP) proposed by Xu et al. (2016). For multi-stage settings with other payoff structures, although we do not have detailed recommendations for implementing the BML approach, we expect it to perform well when using robust Bayesian regression models that are appropriate for the payoff in the data set at hand.

6 Simulation Studies

We conducted two simulation studies that compare implementations of the proposed BML approach with Q-learning in various two-stage settings. Our first simulation study considered a setting with a real-valued payoff, and the nine cases used by Chakraborty et al. (2013) and Laber et al. (2014), which are categorized as non-regular (NR), near-non-regular (NNR) or regular (R). The main purpose of this study was to assess the performance of the BML approach for characterizing uncertainty about $\mu(a_1, d_2^{opt} \,|\, O_1)$ in non-regular settings. We implemented our BML approach using Bayesian linear regression models with non-informative priors, and Q-learning using linear regression models and an m-out-of-n bootstrap (Chakraborty et al., 2013). This comparison isolates the differences between the two approaches for identifying optimal rules and characterizing uncertainty. Our second simulation study considered two other two-stage settings, each with a different payoff structure. The first setting had real-valued initial and subsequent payoffs, and the second setting had binary initial and subsequent payoffs, where only observations with $Y_1 = 0$ move on to stage 2. The main purpose of our second simulation study was to demonstrate the applicability of the BML approach, and secondarily, to investigate the utility of regression models with a nonparametric mean function for identifying optimal decision rules. In both settings of our second simulation study, we contrasted two implementations of the BML approach with comparable versions of Q-learning, (i) the BML approach using generalized linear regression models with non-informative priors ("BML-GLM") versus (ii) Q-learning using generalized linear regression models ("QL-GLM"), and (iii) the BML approach using BART models ("BML-BART") versus (iv) Q-learning using generalized additive regression models ("QL-GAM"). We provide additional details and the results for each simulation study in turn below. We also provide `R` software to implement each method and reproduce the results of these simulation studies, see the Supplementary Material.

6.1 Simulation Study with Non-regular Scenarios

Our first simulation study considered data sets with the sequential structure, $O_1 \rightarrow A_1 \rightarrow O_2 \rightarrow A_2 \rightarrow Y$, where $O_k \in \{-1, 1\}$, $A_k \in \{-1, 1\}$ and $Y \in \mathbb{R}$. We implemented Q-learning by assuming,

$$\begin{aligned}\mu_2(A_2 \,|\, \overline{O}_2; \boldsymbol{\beta}_2) &= \beta_{2,0} + \beta_{2,1}O_1 + \beta_{2,2}A_1 + \beta_{2,3}O_1 \times A_1 + \beta_{2,4}O_2 + \\ &\qquad \beta_{2,5}A_2 + \beta_{2,6}A_2 \times O_1 + \beta_{2,7}A_2 \times A_1 + \beta_{2,8}A_2 \times O_2, \\ \mu(A_1, d_2^{opt} \,|\, O_1; \boldsymbol{\beta}_1) &= \beta_{1,0} + \beta_{1,1}O_1 + \beta_{1,2}A_1 + \beta_{1,3}O_1 \times A_1,\end{aligned}$$

and using the least squares estimates for $\boldsymbol{\beta}_k$, $k = 1, 2$. Q-learning derives the estimate for $\boldsymbol{\beta}_1$ by using the pseudo-outcomes, $\tilde{y}_i = \sup_{a_2 \in \{-1,1\}} \mu_2(a_2 \,|\, \overline{o}_{2,i}; \widehat{\boldsymbol{\beta}})$, as the response variable. To calculate confidence intervals for $\mu_2(A_2 \,|\, \overline{O}_2; \boldsymbol{\beta}_2)$, we used the percentile bootstrap; whereas, to calculate confidence intervals for $\mu(A_1, d_2^{opt} \,|\, O_1; \boldsymbol{\beta}_1)$, we used the fixed $\alpha = 0.05$ version of the m-out-of-n bootstrap proposed by Chakraborty et al. (2013). Similarly, we implemented the BML approach by assuming,

$$\begin{aligned}Y(A_1, A_2) &= \beta_{2,0} + \beta_{2,1}O_1 + \beta_{2,2}A_1 + \beta_{2,3}O_1 \times A_1 + \beta_{2,4}O_2 + \\ &\qquad \beta_{2,5}A_2 + \beta_{2,6}A_2 \times O_1 + \beta_{2,7}A_2 \times A_1 + \beta_{2,8}A_2 \times O_2 + \epsilon_2, \\ Y(A_1, d_2^{opt}) &= \beta_{1,0} + \beta_{1,1}O_1 + \beta_{1,2}A_1 + \beta_{1,3}O_1 \times A_1 + \epsilon_1,\end{aligned}$$

where $\epsilon_k \sim \text{Normal}(0, \sigma_k^2)$ and $p_k(\boldsymbol{\beta}_k, \sigma_k^2) \propto 1/\sigma_k^2$, for $k = 1, 2$. To sample from the stage 1 posterior distribution, we used the BIG sampler described previously. The above methods use the same specifications for the mean functions, but different approaches for estimating and characterizing uncertainty about these mean functions.

Following Laber et al. (2014) and Chakraborty et al. (2013), we generated observations

as follows,

$$\begin{aligned}
&\text{Prob}(O_1 = 1) = 1 - \text{Prob}(O_1 = -1) = 0.5 \\
&\text{Prob}(A_1 = 1 \,|\, O_1) = 1 - \text{Prob}(A_1 = -1 \,|\, O_1) = 0.5 \\
&\text{Prob}(O_2 = 1 \,|\, \overline{A}_1) = 1 - \text{Prob}(O_2 = -1 \,|\, \overline{A}_1) = \text{expit}\{\delta_1 O_1 + \delta_2 A_1\} \\
&\text{Prob}(A_2 = 1 \,|\, \overline{O}_2) = 1 - \text{Prob}(A_2 = -1 \,|\, \overline{O}_2) = 0.5 \\
&Y(A_1, A_2) = \alpha_0 + \alpha_1 O_1 + \alpha_2 A_1 + \alpha_3 O_1 \times A_1 + \alpha_4 O_2 + \\
&\qquad\qquad \alpha_5 A_2 + \alpha_6 A_2 \times O_1 + \alpha_7 A_2 \times A_1 + \alpha_8 A_2 \times O_2 + \epsilon,
\end{aligned}$$

where $\epsilon \sim \text{Normal}(0, 1)$, and $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ are specified in each case to exhibit varying degrees of non-regularity (the values for $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ in each case are reported in the Supplement). Case 1 is an extreme non-regular setting where neither the covariates nor the action at either stage affects the payoff. Case 2 is very similar to Case 1, except that the stage 2 action now has a minuscule effect on the payoff and $d_2^{opt}(\overline{O}_2) = 1$. Case 3 is a non-regular setting where the stage 2 action affects the payoff only for current histories at stage 2 with $A_1 = 1$, and assuming the optimal action at stage 2, the stage 1 action has no effect on the payoff. Case 4 is very similar to Case 3, except that the action at each stage now has a small effect on the payoff, and $d_2^{opt}(\overline{O}_2) = 1$ and $d_1^{opt}(O_1) = -1$. Case 5 is a non-regular setting where the stage 2 action affects the payoff only for current histories without $A_1 = -1$ and $O_2 = -1$, and assuming the optimal action at stage 2, the stage 1 action has no effect on the payoff. Cases 6 and A are regular settings, i.e., where the stage 2 action affects the payoff for all current histories at stage 2, and assuming the optimal action at stage 2, the stage 1 action also affects the payoff. Case B is a non-regular setting where the stage 2 action does not affect the payoff for current histories at stage 2 with $A_1 = -1$, but assuming the optimal action at stage 2, the stage 1 action has a modest effect on the payoff. Case C is very similar to Case B, except that the stage 2 action has a minuscule effect on the payoff for current histories at stage 2 with $A_1 = -1$.

For each case, we applied the two methods to 1000 data sets with either $n = 100$, 300 or

500 observations. For each data set and method, we calculated five metrics at each stage: (1) POA=the proportion of sample observations that would be assigned the optimal action based on the identified optimal decision rule at that stage, (2) Bias=the average difference and (3) RMSE=the square-root of the average squared difference between the estimated mean and the true mean evaluated at each sample observation's current history and each action at that stage, and (4) W95=the average width and (5) C95=coverage of the 95% interval for the mean evaluated at each sample observation's current history and each action at that stage. POA is measure for how well each method identifies the optimal decision rule, Bias and RMSE are measures for how well each method estimates the mean function, and W95 and C95 are measures for how well each method characterizing uncertainty about the mean function.

Table 1 contains the results of our first simulation study for data sets with $n = 300$ observations. The results for data sets with $n = 100$ and 500 observations are reported in the Supplement. Excepting Monte Carlo error associated with the BML method, which can be made arbitrarily small by drawing additional posterior samples, the two methods result in the same estimate for $\mu_2(A_2 \,|\, \overline{O}_2)$, and thus have the same POA, Bias, and RMSE at stage 2. The two methods result in intervals with similar width and coverage rates as well, but the percentile bootstrap for Q-learning is slightly anti-conservative. By contrast, the two methods do not result in the same estimate for $\mu(A_1, d_2^{opt} \,|\, O_1)$, and thus do not necessarily had the same POA, Bias, and RMSE at stage 1. In Case 4 the BML method had a worse POA at stage 1 than the Q-learning method, but in Cases A, B and C it had a slightly better POA at stage 1. The BML method had smaller Bias and RMSE in every NR and NNR case, but slightly larger Bias and RMSE in the two R cases. Both methods resulted in more conservative intervals at stage 1 for cases 1 and 2, which are extreme null and near-null cases where the true regression coefficients are all zero, or nearly zero. Although the BML method tended to provide wider, more conservative intervals in the NR and NNR cases than Q-learning based on the m-out-of-n bootstrap, it provided tighter intervals with the

Table 1: Simulation averages for 1000 data sets each with $n = 300$ observations. POA=proportion of optimal actions assigned by the identified optimal decision rule for observations with $\{\overline{o}_{k,i}\}_{i=1}^{n}$, Bias=bias and RMSE=root-mean-square error of the estimated mean evaluated at $\{\overline{o}_{k,i}\}_{i=1}^{n}$ and $a_k \in \mathcal{A}_k$, and W95=average width and C95=coverage rate of the 95% interval for the mean evaluated at $\{\overline{o}_{k,i}\}_{i=1}^{n}$ and $a_k \in \mathcal{A}_k$.

BML (proposed approach) using linear regression models with non-informative priors

		Stage 1					Stage 2				
Case	Type	POA	Bias	RMSE	W95	C95	POA	Bias	RMSE	W95	C95
1	NR	1.000	0.062	0.135	0.611	0.968	1.000	-0.003	0.169	0.680	0.951
2	NNR	1.000	0.052	0.131	0.611	0.972	0.551	-0.003	0.169	0.680	0.951
3	NR	1.000	0.030	0.140	0.607	0.959	1.000	-0.003	0.169	0.680	0.951
4	NNR	0.650	0.026	0.139	0.607	0.960	0.779	-0.003	0.169	0.680	0.951
5	NR	1.000	0.016	0.146	0.626	0.956	1.000	-0.003	0.168	0.678	0.951
6	R	0.999	-0.010	0.143	0.600	0.949	0.990	-0.003	0.169	0.682	0.949
A	R	0.924	-0.006	0.142	0.598	0.950	0.995	-0.003	0.169	0.682	0.949
B	NR	0.984	0.030	0.141	0.606	0.957	1.000	-0.003	0.169	0.682	0.950
C	NNR	0.982	0.026	0.139	0.606	0.961	0.778	-0.003	0.169	0.682	0.949

Q-Learning using linear regression models

		Stage 1					Stage 2				
Case	Type	POA	Bias	RMSE	W95	C95	POA	Bias	RMSE	W95	C95
1	NR	1.000	0.089	0.148	0.603	0.963	1.000	-0.003	0.169	0.675	0.946
2	NNR	1.000	0.080	0.143	0.603	0.965	0.552	-0.003	0.169	0.675	0.946
3	NR	1.000	0.044	0.147	0.603	0.955	1.000	-0.003	0.169	0.675	0.946
4	NNR	0.703	0.039	0.144	0.603	0.956	0.779	-0.003	0.169	0.675	0.946
5	NR	1.000	0.022	0.147	0.622	0.950	1.000	-0.003	0.168	0.673	0.946
6	R	1.000	-0.003	0.141	0.609	0.950	0.990	-0.003	0.169	0.677	0.946
A	R	0.920	-0.003	0.141	0.600	0.947	0.995	-0.003	0.169	0.677	0.946
B	NR	0.979	0.044	0.147	0.608	0.955	1.000	-0.003	0.168	0.677	0.944
C	NNR	0.975	0.040	0.144	0.609	0.957	0.777	-0.003	0.168	0.677	0.944

correct coverage rate in R cases 6 and A. Similar patterns were present for the other sample sizes that we considered. As the sample size increased, both methods had larger POA and smaller Bias, RMSE, and W95 at both stages. For all cases and sample sizes, the BML method provided intervals with coverage rates near the nominal 95% rate, whereas when $n = 100$, the Q-learning method provided intervals at both stages with coverage rates below the nominal 95% rate. Regarding computational efficiency, the BML method took 1 second per data set with $n = 300$ observations and 2000 posterior samples, whereas the Q-learning

method took 7 seconds per data set with $n = 300$ observations and 2000 bootstrap samples. That is, the BML method was about 7 times faster than the Q-learning method. Because direct sampling from the relevant posterior distributions was feasible in this context, the BML method was very fast.

6.2 Simulation Study with Other Payoff Structures

We conducted a second simulation study that considered two two-stage settings that differ from our first simulation study. In the first setting, the sequential structure is $O_1 \to A_1 \to Y_1 \to A_2 \to Y_2$, where $O_1 \in (-1, 1)$, $A_k \in \{-1, 1\}$ and $Y_k \in \mathbb{R}$, $k = 1, 2$. The key difference between this setting and our first simulation study is the initial payoff, Y_1. After exchanging O_2 with Y_1 in the stage 2 regression model and Q-function, we implemented the BML-GLM and QL-GLM methods similarly as before. We implemented the BML-BART method using the default prior specifications and $m = 40$ regression trees for both models, which we selected after preliminary investigation based on one particular data set. We implemented the QL-GAM method by assuming,

$$\mu_2(A_2 \,|\, \overline{Y}_1) = \psi_{2,1}(O_1 \,|\, A_1) + \psi_{2,2}(O_1 \,|\, A_2) + \phi A_2 \times A_1 + \psi_{2,3}(Y_1 \,|\, A_2),$$
$$\mu(A_1, d_2^{opt} \,|\, O_1) = \psi_{1,1}(O_1 \,|\, A_1),$$

where $\psi_{2,1}$, $\psi_{2,2}$, $\psi_{2,3}$ and $\psi_{1,1}$ are penalized splines. To fit the above QL-GAM method, we used the `mgcv` package in `R`, and to characterize uncertainty, we used the percentile bootstrap.

In the second setting, the sequential structure is $O_1 \to A_1 \to Y_1 \to O_2 \to A_2 \to Y_2$, where $O_k \in (-1, 1)$, $A_k \in \{-1, 1\}$ and $Y_k \in \{0, 1\}$, $k = 1, 2$. Moreover, subjects with $Y_1 = 1$, i.e., subjects that responded in stage 1, did not continue on to stage 2, and thus $Y \in \{0, 1\}$. Because subjects with $Y_1 = 1$ do not provide information for the stage 2 regression model, for each method we used the subset of subjects with $Y_1 = 0$ to fit the stage 2 regression model or Q-function. We implemented the BML-GLM method using logistic regression models, and we used the `BayesLogit` package in `R` for posterior sampling. We implemented

the BML-BART method using probit BART models with the default prior specifications and $m = 20$ regression trees for both models. We specified the binary offset for each probit BART model so that the prior mean was equal to the empirical response rate. Because the pseudo-outcomes for the stage 1 Q-function are a mixture of one's and numerical values between zero and one, we implemented the QL-GLM and QL-GAM methods using quasi-binomial regression models. As far as we are aware, implementing Q-learning with quasi-binomial regression models is novel (cf. Moodie et al., 2014). We used the percentile bootstrap for the QL-GLM method, but this was not feasible for the QL-GAM method, likely because the optimization routine was unstable for a data set with replicate observations and binary response variables.

For both settings, we generated data sets with $n = 300$ observations such that $O_1 \sim \text{Unif}(-1, 1)$ and $\text{Prob}(A_k = 1) = 1 - \text{Prob}(A_k = -1) = 0.5$, $k = 1, 2$. We generated the remaining random variables in each setting, e.g., (Y_1, Y_2) and (Y_1, O_2, Y_2), assuming either (i) linear or (ii) non-linear associations with the preceding variables. We provide the exact data generation models for each *linear* and *non-linear* scenario in the Supplement. For each data set and method, we calculated five metrics at each stage, which we defined previously, POA, Bias, RMSE, W95, and C95. Because the BML-GLM and QL-GLM methods assumed a linear association between the mean and covariates at each stage, we expected these methods to perform well when the data generation model was linear and poorly when it was non-linear. By contrast, we expected the BML-BART and QL-GAM methods to perform slightly worse than the linear methods when data generation model was linear, and much better when it was non-linear.

Table 2 contains the results of our second simulation study. In Scenario 1, with a real-valued payoff that has linear associations with the covariates and actions, the BML-GLM and QL-GLM methods performed very similarly, and as expected, better than the more flexible BML-BART and QL-GAM methods. For example, both methods had slightly larger POA, smaller RMSE and W95, and C95 near the nominal 95% rate. By contrast, in Scenario 2,

Table 2: Simulation averages for 1000 data sets each with $n = 300$ observations. BML-GLM=proposed approach and QL-GLM=Q-learning using generalized linear models, BML-BART=proposed approach using BART models, QL-GAM=Q-learning using generalized additive models. POA=proportion of optimal actions assigned by the identified optimal decision rule for observations with $\{\overline{o}_{k,i}\}_{i=1}^{n_k}$, Bias=bias and RMSE=root-mean-square error of the estimated mean evaluated at $\{\overline{o}_{k,i}\}_{i=1}^{n_k}$ and $a_k \in \mathcal{A}_k$, and W95=average width and C95=coverage rate of the 95% interval for the mean evaluated at $\{\overline{o}_{k,i}\}_{i=1}^{n_k}$ and $a_k \in \mathcal{A}_k$.

Scenario 1: Real-valued payoff with *linear* associations.

	Stage 1					Stage 2				
Method	POA	Bias	RMSE	W95	C95	POA	Bias	RMSE	W95	C95
BML-GLM	0.943	0.000	0.181	0.743	0.951	0.932	-0.001	0.168	0.671	0.952
QL-GLM	0.944	0.010	0.180	0.780	0.957	0.932	-0.001	0.168	0.666	0.947
BML-BART	0.932	-0.039	0.283	1.484	0.988	0.915	-0.001	0.253	1.222	0.984
QL-GAM	0.929	0.014	0.220	1.301	0.974	0.919	-0.001	0.216	1.315	0.982

Scenario 2: Real-valued payoff with *nonlinear* associations.

	Stage 1					Stage 2				
Method	POA	Bias	RMSE	W95	C95	POA	Bias	RMSE	W95	C95
BML-GLM	0.987	-0.333	0.417	0.354	0.319	0.572	0.000	0.451	0.345	0.264
QL-GLM	0.987	-0.320	0.406	0.366	0.307	0.572	0.000	0.451	0.350	0.268
BML-BART	0.989	-0.032	0.114	0.558	0.978	0.956	0.000	0.118	0.433	0.943
QL-GAM	0.992	-0.011	0.092	0.380	0.924	0.977	0.000	0.065	0.331	0.969

Scenario 3: Binary payoff with *linear* associations.

	Stage 1					Stage 2				
Method	POA	Bias	RMSE	W95	C95	POA	Bias	RMSE	W95	C95
BML-GLM	0.858	0.001	0.050	0.197	0.933	0.852	-0.001	0.112	0.380	0.912
QL-GLM	0.861	0.006	0.049	0.181	0.853	0.853	-0.001	0.110	0.419	0.832
BML-BART	0.861	-0.010	0.056	0.315	0.993	0.857	-0.001	0.109	0.480	0.979
QL-GAM	0.821	0.012	0.069	–	–	0.796	-0.001	0.175	–	–

Scenario 4: Binary payoff with *nonlinear* associations.

	Stage 1					Stage 2				
Method	POA	Bias	RMSE	W95	C95	POA	Bias	RMSE	W95	C95
BML-GLM	0.931	-0.084	0.105	0.267	0.788	0.525	0.002	0.224	0.375	0.499
QL-GLM	0.930	-0.075	0.097	0.239	0.674	0.525	0.002	0.224	0.401	0.565
BML-BART	0.924	-0.050	0.085	0.433	0.990	0.855	0.001	0.161	0.505	0.902
QL-GAM	0.891	-0.002	0.079	–	–	0.888	0.002	0.159	–	–

with a real-valued payoff that has nonlinear associations with the covariates and actions, the BML-GLM and QL-GLM methods performed much worse at stage 2 for POA, RMSE, and C95, and at stage 1 for Bias, RMSE, and C95. BML-BART compared with QL-GAM had

slightly worse POA and RMSE, and wider intervals at both stages. Because the true optimal decision rule at stage 1 was $d_1^{opt}(O_1) = \text{sign}(O_1)$, the GLM methods were able to perform well for identification, but not for estimation or characterizing uncertainty. In Scenario 3, with a binary payoff that has linear associations with the covariates and actions, the BML-GLM and QL-GLM methods had similar POA at both stages, but the QL-GLM had a C95 far below the 95% rate at both stages. The BML-BART method had a POA at both stages similar to the correctly specified GLM methods, but large W95 and conservative C95; whereas, the QL-GAM method performed the worst for every measure. In Scenario 4, with a binary payoff that has nonlinear associations with the covariates and actions, as expected, the BML-GLM and QL-GLM performed poorly relative to the more robust BML-BART and QL-GAM methods. BML-BART compared with QL-GAM had a slightly higher POA at stage 1, but a slightly lower POA at stage 2.

Figure 1 depicts some of the posterior information provided by the BML-BART approach. The top panels (1a) and (2a) depict the true mean functions at each stage for Scenario 2, along with the posterior 95% credible intervals at each sample observation's current history and each action for one particular data set. As shown by panels (1a) and (2a), the BML-BART method was able to detect the non-linear association between the mean of the response and the covariates and actions at each stage. The bottom panels (1b) and (2b) depict the posterior optimality probability for $A_1 = 1$ and $A_2 = 1$ at each sample observation's current history, respectively. Current histories marked with "$\times$" that are below (above) the horizontal grey line would be assigned the optimal (sub-optimal) action under the identified optimal decision rule at each stage, and vice versa for those marked with "$\bullet$." As shown by panels (1b) and (2b), current histories that had a posterior optimality probability for $A_k = 1$ near zero or one would be assigned the optimal action at each stage. By contrast, a small proportion of the current histories that had a posterior optimality probability between 0.1 and 0.9 would be assigned the sub-optimal action at each stage. This indicates that the posterior optimality probabilities derived from BML-BART are reliable quantities that could

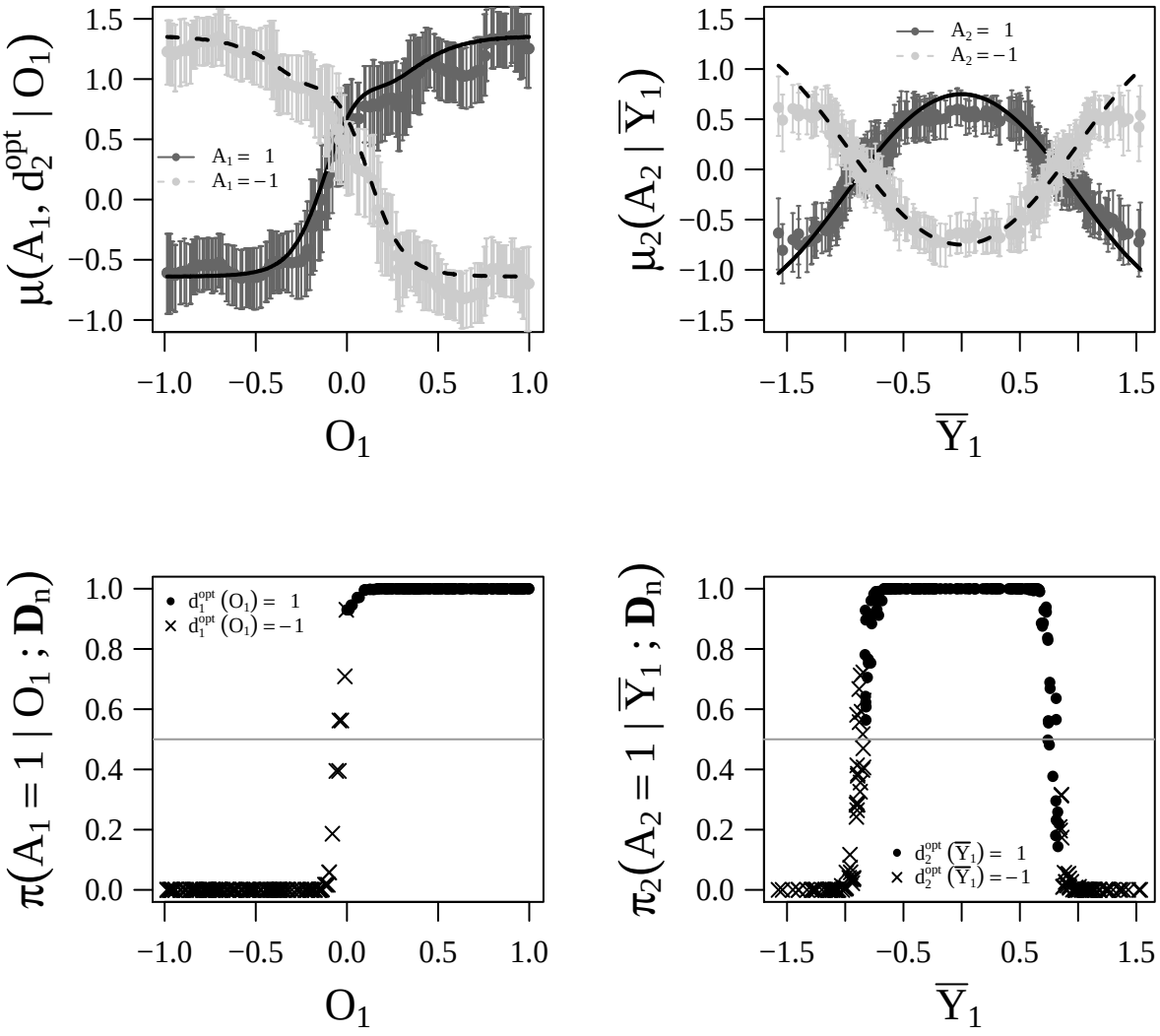

Figure 1: Posterior information from the BML-BART method fit to one particular data set from Scenario 2 of our second simulation study. The top panels (1a) and (2a) depict the mean functions for the two actions at stages 1 and 2, and the posterior 95% credible intervals at $\{o_{1,i}\}_{1=1}^{n}$ and $\{(o_{1,i}, y_{1,i})\}_{1=1}^{n}$, respectively. The bottom panels (1b) and (2b) display the posterior optimality probability for $A_k = 1$ at $\{o_{1,i}\}_{1=1}^{n}$ and $\{(o_{1,i}, y_{1,i})\}_{1=1}^{n}$, respectively. In panels (1b) and (2b), the observations marked with "•" correspond to a true optimal action $A_k = 1$, and those marked with "×" correspond to a true optimal action $A_k = -1$.

be used as an aid for decision-making in practice.

7 Discussion

The proposed BML approach is a general framework that facilitates using Bayesian regression modeling to optimize DTRs. This approach bridges the gap between Bayesian inference and existing machine learning methods, like Q-learning, and thus provides a new avenue for addressing a wide variety of sequential decision-making problems. While the BML approach is extremely general in that it can be applied using any reasonable series of regression models for the problem at hand, we recommend using Bayesian nonparametric regression models to enhance robustness.

In a simulation studies that considered real-valued and binary payoffs, we implemented the BML approach using generalized linear regression models, and more robust BART models. The simulations demonstrated that the BML approach can be used to develop effective and robust methods for optimizing DTRs. Although we focused on settings where the actions were randomly assigned, the proposed BML approach could be used for optimizing DTRs based on an observational data set. In this case, so that the causal effect of the actions may be identified, the regression models will need to account for potential confounding between the actions and the response variable at each stage.

To assist physicians in identifying optimal actions (or treatments), it would be useful to develop tools based on the BML approach, tailored to particular medical settings, that provide the identified optimal action and the posterior optimality probabilities of the relevant actions for the current stage and patient. The physician could use this information when deciding with the patient which action is most appropriate. This procedure makes an allowance for the patient's own desires and the physician's expert knowledge that may not be captured adequately by the regression models and data used to identify the optimal action and calculate the posterior optimality probabilities of the relevant actions. In this way,

the proposed BML approach could engender robust, practical tools for improving medical practice.

SUPPLEMENTARY MATERIAL

Software: `R` software to reproduce the results of the two simulation studies reported in the article. (BML-Software.zip)

Supplement: Supplemental text that contains the additional results for the first simulation study, and the data generation models that we used for the second simulation study, as mentioned in Section 6. (BML-Supplement.pdf)

References

Arjas, E. and O. Saarela (2010). Optimal dynamic regimes: presenting a case for predictive inference. *The International Journal of Biostatistics 6*(2), Article 10.

Bellman, R. (1957). *Dynamic Programming* (1 ed.). Princeton, NJ, USA: Princeton University Press.

Carlin, B. P. and T. A. Louis (2009). *Bayesian Methods for Data Analysis, 3rd edition*. Boca-Raton, FL: Chapman & Hall/CRC Press.

Chakraborty, B. (2011). Dynamic treatment regimes for managing chronic health conditions: a statistical perspective. *American Journal of Public Health 101*(1), 40–45.

Chakraborty, B., E. B. Laber, and Y. Zhao (2013). Inference for optimal dynamic treatment regimes using an adaptive m-out-of-n bootstrap scheme. *Biometrics 69*(3), 714–723.

Chakraborty, B., S. Murphy, and V. Strecher (2010). Inference for non-regular parameters in optimal dynamic treatment regimes. *Statistical Methods in Medical Research 19*(3), 317–343.

Chakraborty, B. and S. A. Murphy (2014). Dynamic treatment regimes. *Annual Review of Statistics and Its Application 1*(1), 447–464.

Chipman, H. A., E. I. George, and R. E. McCulloch (1998). Bayesian CART model search. *Journal of the American Statistical Association 93*(443), 935–948.

Chipman, H. A., E. I. George, and R. E. McCulloch (2010). BART: Bayesian additive regression trees. *The Annals of Applied Statistics 4*(1), 266–298.

Huang, X., S. Choi, L. Wang, and P. F. Thall (2015). Optimization of multi-stage dynamic treatment regimes utilizing accumulated data. *Statistics in Medicine 34*(26), 3424–3443.

Laber, E. B., D. J. Lizotte, M. Qian, W. E. Pelham, and S. A. Murphy (2014). Dynamic treatment regimes: Technical challenges and applications. *Electronic Journal of Statistics 8*(1), 1225–1272.

Lavori, P. W. and R. Dawson (2004). Dynamic treatment regimes: practical design considerations. *Clinical Trials 1*(1), 9–20.

Lee, J., P. F. Thall, Y. Ji, and P. Müller (2015). Bayesian dose-finding in two treatment cycles based on the joint utility of efficacy and toxicity. *Journal of the American Statistical Association 110*(510), 711–722.

Moodie, E. E. M., N. Dean, and Y. R. Sun (2014). Q-learning: Flexible learning about useful utilities. *Statistics in Biosciences 6*(2), 223–243.

Moodie, E. E. M., T. S. Richardson, and D. A. Stephens (2007). Demystifying optimal dynamic treatment regimes. *Biometrics 63*(2), 447–455.

Müller, P., F. A. Quintana, A. Jara, and T. Hanson (2015). *Bayesian Nonparametric Data Analysis*. New York, NY: Springer.

Murphy, S. A. (2003). Optimal dynamic treatment regimes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 65*(2), 331–355.

Murphy, S. A. (2005). A generalization error for Q-learning. *Journal of Machine Learning Research 6*(1), 1073–1097.

Nahum-Shani, I., M. Qian, D. Almirall, W. Pelham, B. Gnagy, G. Fabiano, J. Waxmonsky, J. Yu, and S. Murphy (2012). Q-learning: a data analysis method for constructing adaptive interventions. *Psychological Methods 17*(4), 478–494.

Robins, J. M. (2004). *Proceedings of the Second Seattle Symposium in Biostatistics: Analysis of Correlated Data*, Chapter Optimal Structural Nested Models for Optimal Sequential Decisions, pp. 189–326. New York, NY: Springer New York.

Rubin, D. (1974). Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology 66*(5), 688–701.

Rubin, D. B. (1990). Formal mode of statistical inference for causal effects. *Journal of Statistical Planning and Inference 25*(3), 279 – 292.

Sparapani, R. A., B. R. Logan, R. E. McCulloch, and P. W. Laud (2016). Nonparametric survival analysis using Bayesian additive regression trees (BART). *Statistics in Medicine 35*(16), 2741–2753.

Thall, P. F., R. L. Millikan, and H.-G. Sung (2000). Evaluating multiple treatment courses in clinical trials. *Statisitcs in Medicine 19*(8), 1011–1028.

Thall, P. F., L. H. Wooten, C. J. Logothetis, R. E. Millikan, and N. M. Tannir (2007). Bayesian and frequentist two-stage treatment strategies based on sequential failure times subject to interval censoring. *Statistics in Medicine 26*(26), 4687–4702.

Wang, L., A. Rotnitzky, X. Lin, R. E. Millikan, and P. F. Thall (2012). Evaluation of viable dynamic treatment regimes in a sequentially randomized trial of advanced prostate cancer. *Journal of the American Statistical Association 107*(498), 493–508.

Watkins, C. (1989). *Learning from delayed rewards*. PhD Thesis, University of Cambridge, England.

Xu, Y., P. Mller, A. S. Wahed, and P. F. Thall (2016). Bayesian nonparametric estimation for dynamic treatment regimes with sequential transition times. *Journal of the American Statistical Association 111*(515), 921–950.

Zajonc, T. (2012). Bayesian inference for dynamic treatment regimes: Mobility, equity, and efficiency in student tracking. *Journal of the American Statistical Association 107*(497), 80–92.

Zhang, B., A. A. Tsiatis, E. B. Laber, and M. Davidian (2013). Robust estimation of optimal dynamic treatment regimes for sequential treatment decisions. *Biometrika 100*(3), 681694.

Zhao, Y., M. R. Kosorok, and D. Zeng (2009). Reinforcement learning design for cancer clinical trials. *Statistics in Medicine 28*(26), 3294–3315.

Zhao, Y.-Q., D. Zeng, E. B. Laber, and M. R. Kosorok (2015). New statistical learning methods for estimating optimal dynamic treatment regimes. *Journal of the American Statistical Association 110*(510), 583–598.

Supplement: A Bayesian Machine Learning Approach for Optimizing Dynamic Treatment Regimes

Thomas A. Murray[*,†] (tamurray@mdanderson.org),
Ying Yuan[*,*] (yyuan@mdanderson.org), and
Peter F. Thall[†] (rex@mdanderson.org)
Department of Biostatistics, MD Anderson Cancer Center

April 21, 2017

[*]Yuan's and Murray's research was partially supported by Award Number R01-CA154591 from the National Cancer Institute.

[†]The work of the first three authors was partially funded by NIH/NCI grant 5-R01-CA083932.

Table 1: Data generation model parameter values for the nine cases under consideration in our first simulation study. NR=non-regular, NNR=near non-regular or R=regular.

Case	Type	$\boldsymbol{\delta}$	$\boldsymbol{\alpha}$
1	NR	$(0.5, 0.5)$	$(0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00)$
2	NNR	$(0.5, 0.5)$	$(0.00, 0.00, 0.00, 0.00, 0.00, 0.01, 0.00, 0.00, 0.00)$
3	NR	$(0.5, 0.5)$	$(0.00, 0.00, -.50, 0.00, 0.00, 0.50, 0.00, 0.50, 0.00)$
4	NNR	$(0.5, 0.5)$	$(0.00, 0.00, -.50, 0.00, 0.00, 0.50, 0.00, 0.49, 0.00)$
5	NR	$(1.0, 0.0)$	$(0.00, 0.00, -.50, 0.00, 0.00, 1.00, 0.00, 0.50, 0.50)$
6	R	$(0.1, 0.1)$	$(0.00, 0.00, -.50, 0.00, 0.00, 0.25, 0.00, 0.50, 0.50)$
A	R	$(0.1, 0.1)$	$(0.00, 0.00, -.25, 0.00, 0.00, 0.75, 0.00, 0.50, 0.50)$
B	NR	$(0.0, 0.0)$	$(0.00, 0.00, 0.00, 0.00, 0.00, 0.25, 0.00, 0.25, 0.00)$
C	NNR	$(0.0, 0.0)$	$(0.00, 0.00, 0.00, 0.00, 0.00, 0.25, 0.00, 0.24, 0.00)$

1 Data Generation Models

Below are the data generation models that we used for our second simulation study. In Scenario 1, we generated observations as follows,

$$
\begin{aligned}
O_1 &\sim \text{Unif}\,(-1, 1)\,, \\
A_1 \,|\, O_1 &\sim 2\,[\text{Bernoulli}\,(0.5)] - 1, \\
\boldsymbol{\alpha}_1 &\sim \text{Normal}\,(\mathbf{0}, \text{diag}(0.4, 0.4, 0.4, 0.2))\,, \\
Y_1 \,|\, O_1, A_1; \boldsymbol{\alpha}_1 &\sim \text{Normal}\,(\alpha_{1,0} + \alpha_{1,1} O_1 + \alpha_{1,2} A_1 + \alpha_{1,3} O_1 \times A_1, 1)\,, \\
A_2 \,|\, \overline{Y}_1 &\sim 2\,[\text{Bernoulli}\,(0.5)] - 1, \\
\boldsymbol{\alpha}_2 &\sim \text{Normal}\,(\mathbf{0}, \text{diag}(0.4, 0.4, 0.4, 0.2, 0.4, 0.4, 0.2, 0.2, 0.2))\,, \\
Y_2 \,|\, \overline{Y}_1, A_2; \boldsymbol{\alpha}_2 &\sim \text{Normal}\big(\alpha_{2,0} + \alpha_{2,1} O_1 + \alpha_{2,2} A_1 + \alpha_{2,3} O_1 \times A_1 + \alpha_{2,4} Y_1 + \\
&\qquad \alpha_{2,5} A_2 + \alpha_{2,6} A_2 \times O_1 + \alpha_{2,7} A_2 \times A_1 + \alpha_{2,8} A_2 \times O_2, 1\big).
\end{aligned}
$$

In Scenario 2, we instead generated,

$$
\begin{aligned}
Y_1 \,|\, O_1, A_1 &\sim \text{Normal}\left(A_1[2\text{expit}\{6O_1\} - 1], 0.25^2\right), \\
Y_2 \,|\, \overline{Y}_1, A_2 &\sim \text{Normal}\left(A_2[\cos(\pi Y_1/2) - 0.25], 0.25^2\right),
\end{aligned}
$$

where $\text{expit}\{x\} = \exp\{x\}/(1 + \exp\{x\})$.

In Scenario 3, we generated observations as follows,

$$
\begin{aligned}
O_1 &\sim \text{Unif}\,(-1,1)\,,\\
A_1 \,|\, O_1 &\sim 2\,[\text{Bernoulli}\,(0.5)] - 1,\\
\boldsymbol{\alpha}_1 &\sim \text{Normal}\,(\mathbf{0}, \text{diag}(0.6, 0.6, 0.6, 0.4))\,,\\
Y_1 \,|\, O_1, A_1; \boldsymbol{\alpha}_1 &\sim \text{Bernoulli}\,(\text{expit}\,\{\alpha_{1,0} + \alpha_{1,1}O_1 + \alpha_{1,2}A_1 + \alpha_{1,3}O_1 \times A_1\})\,,\\
\boldsymbol{\gamma} &\sim \text{Normal}\,(\mathbf{0}, \text{diag}(0.6, 0.6, 0.6, 0.4))\,,\\
O_2 \,|\, Y_1 = 0, O_1, A_1; \boldsymbol{\gamma} &\sim 2\big[\text{Beta}\big(5\text{expit}\,\{\gamma_0 + \gamma_1 O_1 + \gamma_2 A_1 + \gamma_3 O_1 \times A_1\}\,,\\
&\qquad 5\,[1 - \text{expit}\,\{\gamma_0 + \gamma_1 O_1 + \gamma_2 A_1 + \gamma_3 O_1 \times A_1\}]\,\big)\big] - 1,\\
A_2 \,|\, \overline{Y}_1 &\sim 2\,[\text{Bernoulli}\,(0.5)] - 1,\\
\boldsymbol{\alpha}_2 &\sim \text{Normal}\,(\mathbf{0}, \text{diag}(0.6, 0.6, 0.6, 0.4, 0.6, 0.6, 0.4, 0.4, 0.4))\,,\\
Y_2 \,|\, \overline{Y}_1, A_2; \boldsymbol{\alpha}_2 &\sim \text{Bernoulli}\big(\text{expit}\big\{\alpha_{2,0} + \alpha_{2,1}O_1 + \alpha_{2,2}A_1 + \alpha_{2,3}O_1 \times A_1 + \alpha_{2,4}Y_1 +\\
&\qquad \alpha_{2,5}A_2 + \alpha_{2,6}A_2 \times O_1 + \alpha_{2,7}A_2 \times A_1 + \alpha_{2,8}A_2 \times O_2\big\}\big).
\end{aligned}
$$

In Scenario 4, we instead generated,

$$
\begin{aligned}
Y_1 \,|\, O_1, A_1 &\sim \text{Bernoulli}\,(A_1 \sin(\pi O_1/1.5) - 0.5)\,,\\
O_2 \,|\, Y_1 = 0, O_1, A_1 &\sim 2\big[\text{Beta}\big(2\text{expit}\,\{0.1(O_1 + A_1)\}\,, 2\,[1 - \text{expit}\,\{0.1(O_1 + A_1)\}]\,\big)\big] - 1,\\
Y_2 \,|\, \overline{Y}_1, A_2 &\sim \text{Bernoulli}\big(\text{expit}\big\{3A_2[(1.5O_2)^2 - 0.25(1.5O_2)^4 - 0.5] - 1\big\}\big).
\end{aligned}
$$

2 Additional Simulation Results

Tables 2 and 3 contain the results of our first simulation study for data sets with $n = 100$ and $n = 500$ observations, respectively.

Table 2: Simulation averages for 1000 data sets each with $n = 100$ observations. POA=proportion of optimal actions assigned by the identified optimal decision rule for observations with $\{\overline{o}_{k,i}\}_{i=1}^{n}$, Bias=bias and RMSE=root-mean-square error of the estimated mean evaluated at $\{\overline{o}_{k,i}\}_{i=1}^{n}$ and $a_k \in \mathcal{A}_k$, and W95=average width and C95=coverage rate of the 95% interval for the mean evaluated at $\{\overline{o}_{k,i}\}_{i=1}^{n}$ and $a_k \in \mathcal{A}_k$.

Proposed approach using linear regression models with non-informative priors

		Stage 1					Stage 2				
Case	Type	POA	Bias	RMSE	W95	C95	POA	Bias	RMSE	W95	C95
1	NR	1.000	0.120	0.243	1.102	0.969	1.000	0.002	0.301	1.207	0.946
2	NNR	1.000	0.110	0.239	1.102	0.969	0.532	0.003	0.301	1.206	0.946
3	NR	1.000	0.061	0.253	1.092	0.955	1.000	0.002	0.301	1.206	0.946
4	NNR	0.644	0.057	0.252	1.092	0.954	0.770	0.002	0.301	1.206	0.946
5	NR	1.000	0.035	0.260	1.121	0.956	1.000	0.002	0.301	1.203	0.946
6	R	0.973	-0.007	0.246	1.093	0.958	0.946	0.002	0.301	1.209	0.948
A	R	0.783	-0.003	0.250	1.096	0.955	0.967	0.002	0.301	1.209	0.948
B	NR	0.832	0.054	0.258	1.105	0.953	0.996	0.003	0.303	1.208	0.947
C	NNR	0.826	0.049	0.256	1.106	0.955	0.761	0.002	0.303	1.208	0.947

Q-Learning using linear regression models

		Stage 1					Stage 2				
Case	Type	POA	Bias	RMSE	W95	C95	POA	Bias	RMSE	W95	C95
1	NR	1.000	0.168	0.267	1.051	0.950	1.000	0.002	0.301	1.191	0.934
2	NNR	1.000	0.158	0.262	1.051	0.955	0.531	0.002	0.301	1.190	0.933
3	NR	1.000	0.086	0.265	1.060	0.937	1.000	0.002	0.301	1.190	0.933
4	NNR	0.694	0.081	0.263	1.061	0.938	0.768	0.002	0.301	1.190	0.933
5	NR	1.000	0.047	0.263	1.094	0.941	1.000	0.002	0.301	1.188	0.932
6	R	0.975	0.014	0.243	1.074	0.940	0.946	0.002	0.301	1.192	0.936
A	R	0.772	0.010	0.248	1.084	0.943	0.968	0.002	0.301	1.192	0.936
B	NR	0.819	0.088	0.266	1.086	0.937	0.996	0.003	0.303	1.192	0.936
C	NNR	0.808	0.083	0.263	1.086	0.938	0.761	0.003	0.303	1.192	0.936

Table 3: Simulation averages for 1000 data sets each with $n = 500$ observations. POA=proportion of optimal actions assigned by the identified optimal decision rule for observations with $\{\overline{o}_{k,i}\}_{i=1}^{n}$, Bias=bias and RMSE=root-mean-square error of the estimated mean evaluated at $\{\overline{o}_{k,i}\}_{i=1}^{n}$ and $a_k \in \mathcal{A}_k$, and W95=average width and C95=coverage rate of the 95% interval for the mean evaluated at $\{\overline{o}_{k,i}\}_{i=1}^{n}$ and $a_k \in \mathcal{A}_k$.

Proposed approach using linear regression models with non-informative priors

		Stage 1					Stage 2				
Case	Type	POA	Bias	RMSE	W95	C95	POA	Bias	RMSE	W95	C95
1	NR	1.000	0.050	0.106	0.470	0.966	1.000	0.001	0.131	0.526	0.948
2	NNR	1.000	0.041	0.103	0.470	0.968	0.557	0.001	0.131	0.525	0.948
3	NR	1.000	0.026	0.109	0.468	0.961	1.000	0.001	0.131	0.525	0.947
4	NNR	0.651	0.022	0.108	0.468	0.960	0.780	0.001	0.131	0.525	0.947
5	NR	1.000	0.016	0.115	0.483	0.951	1.000	0.001	0.130	0.523	0.946
6	R	1.000	-0.001	0.110	0.458	0.952	0.999	0.001	0.130	0.526	0.948
A	R	0.965	0.001	0.110	0.457	0.948	0.999	0.001	0.130	0.526	0.948
B	NR	0.998	0.027	0.110	0.467	0.958	1.000	0.001	0.131	0.527	0.947
C	NNR	0.998	0.022	0.108	0.467	0.959	0.784	0.001	0.131	0.526	0.948

Q-Learning using linear regression models

		Stage 1					Stage 2				
Case	Type	POA	Bias	RMSE	W95	C95	POA	Bias	RMSE	W95	C95
1	NR	1.000	0.071	0.116	0.470	0.965	1.000	0.001	0.131	0.523	0.944
2	NNR	1.000	0.062	0.112	0.470	0.969	0.557	0.001	0.131	0.523	0.944
3	NR	1.000	0.037	0.115	0.468	0.958	1.000	0.001	0.131	0.523	0.944
4	NNR	0.697	0.032	0.112	0.468	0.959	0.781	0.001	0.131	0.523	0.944
5	NR	1.000	0.021	0.116	0.483	0.948	1.000	0.001	0.130	0.521	0.944
6	R	1.000	0.002	0.109	0.469	0.956	0.999	0.001	0.130	0.524	0.946
A	R	0.963	0.002	0.110	0.461	0.948	0.999	0.001	0.130	0.524	0.946
B	NR	0.998	0.037	0.115	0.468	0.956	1.000	0.001	0.131	0.524	0.944
C	NNR	0.998	0.033	0.113	0.468	0.958	0.784	0.001	0.131	0.524	0.944

R Software for "A Bayesian Machine Learning Approach for Optimizing Dynamic Treatment Regimes"

Thomas A. Murray[*,†] (tamurray@mdanderson.org),
Ying Yuan[*,*] (yyuan@mdanderson.org), and
Peter F. Thall[†] (rex@mdanderson.org)
Department of Biostatistics, MD Anderson Cancer Center

June 5, 2017

[*]Yuan's and Murray's research was partially supported by Award Number R01-CA154591 from the National Cancer Institute.
[†]The work of the first three authors was partially funded by NIH/NCI grant 5-R01-CA083932.

readme.txt

This folder contains R programs that faciliate reproducing the simulation study results reported in Murray et al. (2017), "A Bayesian Machine Learning Approach for Optimizing Dynamic Treatment Regimes."

Each file described below and has annotations thoughout.

File, Purpose

simulation1-functions.R, Contains the functions to implement the two methods compared in our first simulation study.

simulation1.R, Reproduces our first simulation study. Depends on simulation1-functions.R

simulation2-real-functions.R, Contains the functions to implement the four methods compared in scenarios 1 and 2 of our second simulation study, i.e., BML-GLM, BML-BART, QL-GLM and QL-GAM.

simulation2-real-linear-associations.R, Reproduces scenario 1 of our second simulation study. Depends on simulation2-real-functions.R

simulation2-real-nonlinear-associations.R, Reproduces scenario 2 of our second simulation study. Depends on simulation2-real-functions.R

simulation2-binary-functions.R, Contains the functions to implement the four methods compared in scenarios 3 and 4 of our second simulation study, i.e

., BML–GLM, BML–BART, QL–GLM and QL–GAM.

simulation2-binary-linear-associations.R, Reproduces scenario 3 of our second simulation study. Depends on simulation2-binary-functions.R

simulation2-binary-nonlinear-associations.R, Reproduces scenario 4 of our second simulation study. Depends on simulation2-binary-functions.R

simulation1-functions.R

```
### Bayesian Machine Learning using Linear Regression Models with Improper
    Priors, i.e., p(beta_k,sigma_k) \propto 1/sigma_k^2
library(MASS) #for mvrnorm function
bml.glm = function(dataset,nsamps=2000){
  # Data
  n = nrow(dataset$data); o1 = dataset$data[,"o1"]; a1 = dataset$data[,"a1"]
  o2 = dataset$data[,"o2"]; a2 = dataset$data[,"a2"]; y = dataset$data[,"y"]

  # Construct design matrices for stage 2 and stage 1 models
  X2 = cbind(1,o1,a1,a1*o1,o2,a2,a2*o1,a2*a1,a2*o2); X1 = cbind(1,o1,a1,a1*o1)
      ; q2 = ncol(X2); q1 = ncol(X1)
  X2.pred = rbind(cbind(cbind(1,o1,a1,a1*o1,o2),cbind(1,o1,a1,o2)),cbind(cbind
      (1,o1,a1,a1*o1,o2),-cbind(1,o1,a1,o2)))
  X1.pred = rbind(cbind(cbind(1,o1),cbind(1,o1)),cbind(cbind(1,o1),-cbind(1,o1
      )))

  # Pre-computations
  X2tX2.inv = solve(t(X2)%*%X2); beta2.hat = X2tX2.inv%*%t(X2)%*%y; X1tX1.inv
      = solve(t(X1)%*%X1)

  # Sample G realizations of stage 2 parameter from its posterior distribution
  sigma2.sq = 1/rgamma(nsamps,shape=(n-q2)/2,rate=t(y-X2%*%beta2.hat)%*%(y-X2%
      *%beta2.hat)/2)
  beta2 = t(sapply(1:nsamps, function(samp) mvrnorm(1,beta2.hat,sigma2.sq[samp
      ]*X2tX2.inv)))

  # Calculate Stage 2 Summary Statistics
  mu2 = as.vector(X2.pred%*%dataset$beta2)
  mu2.samps = X2.pred%*%t(beta2)
  mu2.hat = rowMeans(mu2.samps);
  mu2.low = apply(mu2.samps,1,function(x) quantile(x,0.025));
```

```
mu2.high = apply(mu2.samps,1,function(x) quantile(x,0.975))
POA = mean(sign(mu2.hat[1:n]-mu2.hat[n+1:n])==sign(mu2[1:n]-mu2[n+1:n]) |
    mu2[1:n]==mu2[n+1:n])
bias = mean(mu2.hat-mu2)
RMSE = sqrt(mean((mu2.hat-mu2)**2))
w95 = mean(mu2.high-mu2.low)
c95 = mean(mu2.low < mu2 & mu2 < mu2.high)
stats2 = c(POA, bias ,RMSE, w95, c95)

# Sample G realizations of the stage 1 parameter from its poseterior
    distribution using the BIG sampler
sigma1.sq = rep(NA, nsamps); beta1 = matrix(NA, nsamps, q1); d2.opt.samps =
    sample(nsamps)
for(samp in 1:nsamps){
  ### Step 1. Impute payoffs under the optimal stage 2 decision rule from
      their posterior predictive distributions
  # Step 1. Sample the optimal stage 2 decision rule and determine the
      optimal stage 2 treatment for each patient
  a2.opt = sign(cbind(1,o1,a1,o2)%*%beta2[d2.opt.samps[samp],6:9])
  X2.opt = cbind(1,o1,a1,a1*o1,o2,a2.opt,a2.opt*o1,a2.opt*a1,a2.opt*o2)

  # Step 2. Sample missing payoffs under the sampled optimal stage 2 actions
  y2.opt = cbind(y+X2.opt%*%beta2[samp,]-X2%*%beta2[samp,])

  #Step 2. Sample stage 1 parameters from their full conditional posterior
      distribution
  beta1.hat = X1tX1.inv%*%t(X1)%*%y2.opt
  sigma1.sq[samp] = 1/rgamma(1,shape=(n-q1)/2,rate=t(y2.opt-X1%*%beta1.hat)%
      *%(y2.opt-X1%*%beta1.hat)/2)
  beta1[samp,] = mvrnorm(1,beta1.hat,sigma1.sq[samp]*X1tX1.inv)
}
```

```
  # Calculate Stage 1 Summary Statistics
  mu1 = as.vector(X1.pred%*%dataset$beta1)
  mu1.samps = X1.pred%*%t(beta1)
  mu1.hat = rowMeans(mu1.samps);
  mu1.low = apply(mu1.samps,1,function(x) quantile(x,0.025));
  mu1.high = apply(mu1.samps,1,function(x) quantile(x,0.975))
  POA = mean(sign(mu1.hat[1:n]-mu1.hat[n+1:n])==sign(mu1[1:n]-mu1[n+1:n]) |
      mu1[1:n]==mu1[n+1:n])
  bias = mean(mu1.hat-mu1)
  RMSE = sqrt(mean((mu1.hat-mu1)**2))
  w95 = mean(mu1.high-mu1.low)
  c95 = mean(mu1.low < mu1 & mu1 < mu1.high)
  stats1 = c(POA,bias,RMSE,w95,c95)

  # Combine Stage 2 and Stage 1 Summary Statistics
  stats = round(c(stats1,stats2),4)

  return(stats)
}



### Q-Learning using Linear Regression Models and the m-out-of-n bootstrap
ql.glm = function(dataset,nboot=2000,fixedXi=0.05){
  # data
  n = nrow(dataset$data); o1 = dataset$data[,"o1"]; a1 = dataset$data[,"a1"]
  o2 = dataset$data[,"o2"]; a2 = dataset$data[,"a2"]; y = dataset$data[,"y"]

  # construct design matrices for stage 2 and stage 1 models
  X2 = cbind(1,o1,a1,a1*o1,o2,a2,a2*o1,a2*a1,a2*o2); X1 = cbind(1,o1,a1,a1*o1)
      ; q2 = ncol(X2); q1 = ncol(X1)
```

```
X2.pred = rbind(cbind(cbind(1,o1,a1,a1*o1,o2),cbind(1,o1,a1,o2)),cbind(cbind
    (1,o1,a1,a1*o1,o2),-cbind(1,o1,a1,o2)))
X1.pred = rbind(cbind(cbind(1,o1),cbind(1,o1)),cbind(cbind(1,o1),-cbind(1,o1
    )))

# Stage 2 Parameter Estimation and Percentile Bootstrap
X2tX2.inv = solve(t(X2)%*%X2); beta2.hat = X2tX2.inv%*%t(X2)%*%y
mu2.hat = as.vector(X2.pred%*%beta2.hat)
bootest2 <- matrix(NA,nboot,2*n); g=0
while(g<nboot){
  index <- sample(n,n,replace=TRUE); beta2.hat.temp <- as.vector(coef(lm(y[
      index]~X2[index,]-1)))
  if(sum(is.na(beta2.hat.temp))==0){
    g <- g+1; bootest2[g,] <- 2*mu2.hat-as.vector(X2.pred%*%cbind(beta2.hat.
        temp))
  }
}
mu2.low = apply(bootest2,2,function(x) quantile(x,0.025)); mu2.high = apply(
    bootest2,2,function(x) quantile(x,0.975));

# Calculate Stage 2 Summary Statistics
mu2 = as.vector(X2.pred%*%dataset$beta2)
POA = mean(sign(mu2.hat[1:n]-mu2.hat[n+1:n])==sign(mu2[1:n]-mu2[n+1:n]) |
    mu2[1:n]==mu2[n+1:n])
bias = mean(mu2.hat-mu2)
RMSE = sqrt(mean((mu2.hat-mu2)**2))
w95 = mean(mu2.high-mu2.low)
c95 = mean(mu2.low < mu2 & mu2 < mu2.high)
stats2 = c(POA,bias,RMSE,w95,c95)

# Stage 1 Parameter Estimation and m-out-of-n Percentile Bootstrap
a2.opt = sign(cbind(1,o1,a1,o2)%*%beta2.hat[6:9]); X2.opt = cbind(1,o1,a1,a1
```

```
    *o1,o2,a2.opt,a2.opt*o1,a2.opt*a1,a2.opt*o2)
y.tilde = as.vector(X2.opt%*%beta2.hat); beta1.hat = solve(t(X1)%*%X1)%*%t(
    X1)%*%y.tilde
mu1.hat = as.vector(X1.pred%*%beta1.hat)
# Determine m
Z2 <- diag(y-as.vector(X2%*%beta2.hat))%*%X2%*%(n*X2tX2.inv)/sqrt(n-q2);
    Cov2 <- t(Z2)%*%Z2
p <- mean(abs(cbind(a2,a2*o1,a2*a1,a2*o2)%*%beta2.hat[6:9])/sqrt(diag(cbind
    (1,o1,a1,o2)%*%Cov2[6:9,6:9]%*%rbind(1,o1,a1,o2))/n) <= qnorm(0.9995))
m <- ceiling(n^(1-p*(fixedXi/(1+fixedXi))))
# m-out-of-n percentile bootstrap
bootest1 <- matrix(NA,nboot,2*n); g = 0
while(g<nboot){
  index <- sample(n,m,replace = TRUE); beta2.hat.temp <- as.vector(coef(lm(y
      [index]~X2[index,]-1)))
  if(sum(is.na(beta2.hat.temp))==0){
    y.tilde.temp <- cbind(1,o1,a1,a1*o1,o2)[index,]%*%beta2.hat.temp[1:5] +
        abs(cbind(1,o1,a1,o2)[index,]%*%beta2.hat.temp[6:9])
    beta1.hat.temp <- as.vector(coef(lm(y.tilde.temp~X1[index,]-1)))
    if(sum(is.na(beta1.hat.temp))==0){
      g <- g+1; bootest1[g,] <- 2*mu1.hat-as.vector(X1.pred%*%cbind(beta1.
          hat.temp))
    }
  }
}
mu1.low = apply(bootest1,2,function(x) quantile(x,0.025)); mu1.high = apply(
    bootest1,2,function(x) quantile(x,0.975));

#Calculate Stage 1 Summary Statistics
mu1 = as.vector(X1.pred%*%dataset$beta1)
POA = mean(sign(mu1.hat[1:n]-mu1.hat[n+1:n])==sign(mu1[1:n]-mu1[n+1:n]) |
    mu1[1:n]==mu1[n+1:n])
```

```
  bias = mean(mu1.hat-mu1)
  RMSE = sqrt(mean((mu1.hat-mu1)**2))
  w95 = mean(mu1.high-mu1.low)
  c95 = mean(mu1.low < mu1 & mu1 < mu1.high)
  stats1 = c(POA, bias ,RMSE, w95, c95)

  # Combine Stats
  stats = round(c(stats1 , stats2) ,4)

  return(stats)
}
```

simulation1.R

```
##############################################################################
###  This Program Reproduces the Results of Our First Simulation Study  ###
##############################################################################
source('simulation1-functions.R') #Load Functions

### Numerically Determine True beta1 for each case
#memory.limit(10000)
#scens = c("1","2","3","4","5","6","A","B","C")
#for(scen in scens){
#  if(scen=="1"){ beta2=c(0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00); delta
    =c(0.5,0.5) }
#  if(scen=="2"){ beta2=c(0.00,0.00,0.00,0.00,0.00,0.01,0.00,0.00,0.00); delta
    =c(0.5,0.5) }
#  if(scen=="3"){ beta2=c(0.00,0.00,-.50,0.00,0.00,0.50,0.00,0.50,0.00); delta
    =c(0.5,0.5) }
#  if(scen=="4"){ beta2=c(0.00,0.00,-.50,0.00,0.00,0.50,0.00,0.49,0.00); delta
    =c(0.5,0.5) }
#  if(scen=="5"){ beta2=c(0.00,0.00,-.50,0.00,0.00,1.00,0.00,0.50,0.50); delta
    =c(1.0,0.0) }
#  if(scen=="6"){ beta2=c(0.00,0.00,-.50,0.00,0.00,0.25,0.00,0.50,0.50); delta
    =c(0.1,0.1) }
#  if(scen=="A"){ beta2=c(0.00,0.00,-.25,0.00,0.00,0.75,0.00,0.50,0.50); delta
    =c(0.1,0.1) }
#  if(scen=="B"){ beta2=c(0.00,0.00,0.00,0.00,0.00,0.25,0.00,0.25,0.00); delta
    =c(0.0,0.0) }
#  if(scen=="C"){ beta2=c(0.00,0.00,0.00,0.00,0.00,0.25,0.00,0.24,0.00); delta
    =c(0.0,0.0) }
#
#  O1 = 2*rbinom(10000000,1,0.5)-1
#  A1 = 2*rbinom(10000000,1,0.5)-1
#  O2 = 2*rbinom(10000000,1,1/(1+exp(-cbind(O1,A1)%*%delta)))-1
```

```
#   A2 = as.vector(sign(cbind(1,O1,A1,O2)%*%beta2[6:9]))
#   Y = cbind(1,O1,A1,O1*A1,O2,A2,A2*O1,A2*A1,A2*O2)%*%beta2
#   beta1 = round(as.vector(lm(Y~O1+A1+O1:A1)$coef),3)
#   print(scen); print(beta1)
#}
#rm(O1,A1,O2,A2,Y)


### Function to Generate Datasets
get.dataset = function(n,scen){

  # Data Generation Parameter Specifications
  if(scen=="1"){ beta1=c(0.000,0.000,0.000,0.000); beta2=c
      (0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00); delta=c(0.5,0.5) }
  if(scen=="2"){ beta1=c(0.010,0.000,0.000,0.000); beta2=c
      (0.00,0.00,0.00,0.00,0.00,0.01,0.00,0.00,0.00); delta=c(0.5,0.5) }
  if(scen=="3"){ beta1=c(0.500,0.000,0.000,0.000); beta2=c
      (0.00,0.00,-.50,0.00,0.00,0.50,0.00,0.50,0.00); delta=c(0.5,0.5) }
  if(scen=="4"){ beta1=c(0.500,0.000,-.010,0.000); beta2=c
      (0.00,0.00,-.50,0.00,0.00,0.50,0.00,0.49,0.00); delta=c(0.5,0.5) }
  if(scen=="5"){ beta1=c(1.000,0.231,0.000,0.000); beta2=c
      (0.00,0.00,-.50,0.00,0.00,1.00,0.00,0.50,0.50); delta=c(1.0,0.0) }
  if(scen=="6"){ beta1=c(0.644,0.006,-.369,0.019); beta2=c
      (0.00,0.00,-.50,0.00,0.00,0.25,0.00,0.50,0.50); delta=c(0.1,0.1) }
  if(scen=="A"){ beta1=c(0.881,0.019,0.144,0.006); beta2=c
      (0.00,0.00,-.25,0.00,0.00,0.75,0.00,0.50,0.50); delta=c(0.1,0.1) }
  if(scen=="B"){ beta1=c(0.250,0.000,0.250,0.000); beta2=c
      (0.00,0.00,0.00,0.00,0.00,0.25,0.00,0.25,0.00); delta=c(0.0,0.0) }
  if(scen=="C"){ beta1=c(0.250,0.000,0.240,0.000); beta2=c
      (0.00,0.00,0.00,0.00,0.00,0.25,0.00,0.24,0.00); delta=c(0.0,0.0) }

  # Sample Observations
```

```
  o1 = 2*rbinom(n,1,0.5)-1                                                    #
      baseline biomarker
  a1 = 2*rbinom(n,1,0.5)-1                                                    #
      treatment 1
  o2 = 2*rbinom(n,1,1/(1+exp(-cbind(o1,a1)%*%delta)))-1                       #
      interim response
  a2 = 2*rbinom(n,1,0.5)-1                                                    #
      treatment 2
  y = rnorm(n,cbind(1,o1,a1,o1*a1,o2,a2,a2*o1,a2*a1,a2*o2)%*%beta2,1) # payoff

  return(list(data=cbind(o1,a1,o2,a2,y), beta1=beta1, beta2=beta2))
}


#################################
### Conduct Simulation Study ###
#################################
niters = 1000          # Number of Simulated Datasets
ns = c(100,300,500)    # Sample Sizes under consideration
scens = c("1","2","3","4","5","6","A","B","C") # Scenarios under consideration
nsamps = nboot = 2000 # Number of Posterior and Bootstrap Samples

### We recommend running the following parallely on a server
### This code writes the results to the folder "Results" that needs to be
    created in the working directory
for(n in ns){
  print(paste("Starting simulations with n=",n,sep=""))
  for(scen in scens){
    set.seed(1985); datasets <- lapply(1:niters,function(iter) get.dataset(n=n
        ,scen=scen))
    stats.bml <- t(sappy(1:niters,function(iter) bml.glm(dataset=datasets[[
        iter]],nsamps=nsamps)))
```

```
    write.table(stats.bml, file=paste('Results',n,'/BML-Scen',scen,'.txt',sep=
        ''),sep=",",row.names=FALSE, col.names=FALSE)
    stats.ql  <- t(sapply(1:niters,function(iter) ql.glm(dataset=datasets[[
        iter]],nboot=nboot)))
    write.table(stats.ql, file=paste('Results',n,'/QL-Scen',scen,'.txt',sep="")
        ,sep=",",row.names=FALSE, col.names=FALSE)
    print(paste("Scenario",scen,"is done",sep=" "))
  }
  print(paste("Finished simulations with n=",n,sep=" "))
}

##########################
### Simulation Results ###
##########################
library(xtable)
for(n in ns){
  for(method in c("BML","QL")){
    res = NULL
    for(scen in scens){
      if(method=="BML") stats = read.table(paste('Results',n,'/BML-Scen',scen,
          '.txt',sep=""),header=FALSE,sep=",")
      if(method=="QL")  stats = read.table(paste('Results',n,'/QL-Scen',scen,'
          .txt',sep=""),header=FALSE,sep=",")
      res <- rbind(res,colMeans(stats))
    }
    res = cbind(c("NR","NNR","NR","NNR","NR","R","R","NR","NNR"),round(res,3))
    rownames(res) = scens; colnames(res) = c("Type",rep(c("POA","Bias","RMSE",
        "W95","C95"),2))
    print(paste("Simulation Results for ",method," with n=",n,sep=""))
    print(xtable(res,digits=3))
  }
}
```

simulation2-real-functions.R

```
### Bayesian Machine Learning using Linear Regression Models with Improper
    Priors, i.e., p(beta_k,sigma_k) \propto 1/sigma_k^2
library(MASS) #for mvrnorm function
bml.glm = function(dataset,nsamps=2000){
  # data
  n = nrow(dataset$data); o1 = dataset$data[,"o1"]; a1 = dataset$data[,"a1"]
  y1 = dataset$data[,"y1"]; a2 = dataset$data[,"a2"]; y2 = dataset$data[,"y2"]

  # construct design matrices for stage 2 and stage 1 models
  X2 = cbind(1,o1,a1,a1*o1,y1,a2,a2*o1,a2*a1,a2*y1); X1 = cbind(1,o1,a1,a1*o1)
      ; q2 = ncol(X2); q1 = ncol(X1)
  X2.pred = rbind(cbind(cbind(1,o1,a1,a1*o1,y1),cbind(1,o1,a1,y1)),cbind(cbind
      (1,o1,a1,a1*o1,y1),-cbind(1,o1,a1,y1)))
  X1.pred = rbind(cbind(cbind(1,o1),cbind(1,o1)),cbind(cbind(1,o1),-cbind(1,o1
      )))

  # Pre-processing Values
  X2tX2.inv = solve(t(X2)%*%X2); beta2.hat = X2tX2.inv%*%t(X2)%*%y2; X1tX1.inv
       = solve(t(X1)%*%X1)

  # Sample stage 2 parameter from its posterior distribution
  sigma2.sq = 1/rgamma(nsamps,shape=(n-q2)/2,rate=t(y2-X2%*%beta2.hat)%*%(y2-
      X2%*%beta2.hat)/2)
  beta2.samps = t(sapply(1:nsamps, function(samp) mvrnorm(1,beta2.hat,sigma2.
      sq[samp]*X2tX2.inv)))

  # Calculate Stage 2 Summary Statistics
  mu2 = dataset$mu2
  mu2.samps = X2.pred%*%t(beta2.samps)
  mu2.hat = rowMeans(mu2.samps)
  mu2.low = apply(mu2.samps,1,function(x) quantile(x,0.025))
```

```
mu2.high = apply(mu2.samps,1,function(x) quantile(x,0.975))
POA = mean(sign(mu2.hat[1:n]-mu2.hat[n+1:n])==sign(mu2[1:n]-mu2[n+1:n]))
bias = mean(mu2.hat-mu2)
RMSE = sqrt(mean((mu2.hat-mu2)**2))
w95 = mean(mu2.high-mu2.low)
c95 = mean(mu2.low < mu2 & mu2 < mu2.high)
stats2 = c(POA,bias,RMSE,w95,c95)

# Sample the stage 1 parameter from its poseterior distribution using the
    BIG sampler
sigma1.sq = rep(NA,nsamps); beta1.samps = matrix(NA,nsamps,q1); d2.opt.samps
    = sample(nsamps)
for(samp in 1:nsamps){
  # Step 1. Sample the optimal stage 2 decision rule and determine the
      optimal stage 2 treatment for each patient
  a2.opt = sign(cbind(1,o1,a1,y1)%*%beta2.samps[d2.opt.samps[samp],6:9])
  X2.opt = cbind(1,o1,a1,a1*o1,y1,a2.opt,a2.opt*o1,a2.opt*a1,a2.opt*y1)

  # Step 2. Sample missing subsequent payoff under the sampled optimal stage
      2 actions
  y2.opt = cbind(y2+X2.opt%*%beta2.samps[samp,]-X2%*%beta2.samps[samp,])

  # Step 3. Sample parameters from the full conditional stage 1 posterior
      distribution
  beta1.hat.temp = X1tX1.inv%*%t(X1)%*%(y1+y2.opt)
  sigma1.sq[samp] = 1/rgamma(1,shape=(n-q1)/2,rate=t((y1+y2.opt)-X1%*%beta1.
      hat.temp)%*%((y1+y2.opt)-X1%*%beta1.hat.temp)/2)
  beta1.samps[samp,] = mvrnorm(1,beta1.hat.temp,sigma1.sq[samp]*X1tX1.inv)
}

# Calculate Stage 1 Summary Statistics
mu1 = dataset$mu1
```

```
  mu1.samps = X1.pred%*%t(beta1.samps)
  mu1.hat = rowMeans(mu1.samps)
  mu1.low = apply(mu1.samps,1,function(x) quantile(x,0.025))
  mu1.high = apply(mu1.samps,1,function(x) quantile(x,0.975))
  POA = mean(sign(mu1.hat[1:n]-mu1.hat[n+1:n])==sign(mu1[1:n]-mu1[n+1:n]))
  bias = mean(mu1.hat-mu1)
  RMSE = sqrt(mean((mu1.hat-mu1)**2))
  w95 = mean(mu1.high-mu1.low)
  c95 = mean(mu1.low < mu1 & mu1 < mu1.high)
  stats1 = c(POA, bias ,RMSE, w95, c95)

  # Combine Stage 2 and Stage 1 Summary Statistics
  stats = round(c(stats1,stats2),4)

  return(stats)
}



### Bayesian Machine Learning using BART models, prior is the prior
    probability that E[Y|X] \in [min(Y_i:i=1,...,n),max(Y_i:i=1,...,n)]
library(BayesTree)
bml.bart = function(dataset,nsamps=2000,warmup=100,ntree=40,prior=0.95){
  # data
  n = nrow(dataset$data); o1 = dataset$data[,"o1"]; a1 = dataset$data[,"a1"]
  y1 = dataset$data[,"y1"]; a2 = dataset$data[,"a2"]; y2 = dataset$data[,"y2"]

  # Sample nsamps realizations of stage 2 parameter from its posterior
      distribution
  X2 = cbind(o1,a1,y1,a2); X2.pred = rbind(cbind(o1,a1,y1,1),cbind(o1,a1,y1
      ,-1)) #Predictions
  mu2.samps <- bart(X2,y2,X2.pred,ntree=ntree,k=qnorm((prior+1)/2),ndpost=
```

```
    nsamps, nskip=warmup, keeptrainfits=FALSE, verbose=FALSE)$yhat.test

# Calculate Stage 2 Summary Statistics
mu2 = dataset$mu2
mu2.hat = colMeans(mu2.samps)
mu2.low = apply(mu2.samps,2,function(x) quantile(x,0.025))
mu2.high = apply(mu2.samps,2,function(x) quantile(x,0.975))
POA = mean(sign(mu2.hat[1:n]-mu2.hat[n+1:n])==sign(mu2[1:n]-mu2[n+1:n]))
bias = mean(mu2.hat-mu2)
RMSE = sqrt(mean((mu2.hat-mu2)**2))
w95 = mean(mu2.high-mu2.low)
c95 = mean(mu2.low < mu2 & mu2 < mu2.high)
stats2 = c(POA, bias, RMSE, w95, c95)

# Sample G realizations of the stage 1 parameter from its poseterior
    distribution using the BIG sampler
X1 = cbind(o1,a1); X1.pred = rbind(cbind(o1,1),cbind(o1,-1)); mu1.samps =
    matrix(NA,nrow=nsamps,ncol=2*n); d2.opt.samps = sample(nsamps)
for(samp in 1:nsamps){
  # Step 1. Sample the optimal stage 2 rule and determine optimal stage 2
      treatment for each patient under this rule
  a2.opt = sign(mu2.samps[d2.opt.samps[samp],1:n]-mu2.samps[d2.opt.samps[
      samp],n+1:n])

  # Step 2. Sample missing payoffs under the sampled optimal stage 2 actions
  y2.opt = (y2+((a2.opt==1)*mu2.samps[samp,1:n]+(a2.opt==-1)*mu2.samps[samp,
      n+1:n])-((a2==1)*mu2.samps[samp,1:n]+(a2==-1)*mu2.samps[samp,n+1:n]))

  # Step 2. Sample parameters from the full conditional stage 1 posterior
      distribution
  mu1.samps[samp,] <- bart(X1,y1+y2.opt,X1.pred,ntree=ntree,k=qnorm((prior
      +1)/2),ndpost=2,nskip=warmup, keeptrainfits=FALSE, verbose=FALSE)$yhat.
```

```
        test[2,]
    }

    # Calculate Stage 1 Summary Statistics
    mu1 = dataset$mu1
    mu1.hat = colMeans(mu1.samps)
    mu1.low = apply(mu1.samps,2,function(x) quantile(x,0.025))
    mu1.high = apply(mu1.samps,2,function(x) quantile(x,0.975))
    POA = mean(sign(mu1.hat[1:n]-mu1.hat[n+1:n])==sign(mu1[1:n]-mu1[n+1:n]))
    bias = mean(mu1.hat-mu1)
    RMSE = sqrt(mean((mu1.hat-mu1)**2))
    w95 = mean(mu1.high-mu1.low)
    c95 = mean(mu1.low < mu1 & mu1 < mu1.high)
    stats1 = c(POA, bias ,RMSE, w95, c95)

    # Combine Stage 2 and Stage 1 Summary Statistics
    stats = round(c(stats1,stats2),4)

    return(stats)
}




### Q-Learning using Linear Regression Models and the m-out-of-n bootstrap
ql.glm = function(dataset,nboot=2000,fixedXi=0.05){
    # data
    n = nrow(dataset$data); o1 = dataset$data[,"o1"]; a1 = dataset$data[,"a1"]
    y1 = dataset$data[,"y1"]; a2 = dataset$data[,"a2"]; y2 = dataset$data[,"y2"]

    # construct design matrices for stage 2 and stage 1 models
    X2 = cbind(1,o1,a1,a1*o1,y1,a2,a2*o1,a2*a1,a2*y1); X1 = cbind(1,o1,a1,a1*o1)
```

```
    ; q2 = ncol(X2); q1 = ncol(X1)
X2.pred = rbind(cbind(cbind(1,o1,a1,a1*o1,y1),cbind(1,o1,a1,y1)),cbind(cbind
    (1,o1,a1,a1*o1,y1),-cbind(1,o1,a1,y1)))
X1.pred = rbind(cbind(cbind(1,o1),cbind(1,o1)),cbind(cbind(1,o1),-cbind(1,o1
    )))

# Implement Method
X2tX2.inv = solve(t(X2)%*%X2); beta2.hat = X2tX2.inv%*%t(X2)%*%y2
mu2.hat = as.vector(X2.pred%*%beta2.hat)
a2.opt = sign(cbind(1,o1,a1,y1)%*%beta2.hat[6:9])
X2.opt = cbind(1,o1,a1,a1*o1,y1,a2.opt,a2.opt*o1,a2.opt*a1,a2.opt*y1)
y.tilde = y1+as.vector(X2.opt%*%beta2.hat)
X1tX1.inv = solve(t(X1)%*%X1); beta1.hat = X1tX1.inv%*%t(X1)%*%y.tilde
mu1.hat = as.vector(X1.pred%*%beta1.hat)

### pecentile bootstrap for confidence intervals on mu2
bootest2 <- matrix(NA,nboot,2*n); g=0
while(g<nboot){
  index <- sample(n,n,replace=TRUE); beta2.hat.temp <- as.vector(coef(lm(y2[
      index]~X2[index,]-1)))
  if(sum(is.na(beta2.hat.temp))==0){
    g <- g+1; bootest2[g,] <- 2*mu2.hat-as.vector(X2.pred%*%cbind(beta2.hat.
        temp))
  }
}
mu2.low = apply(bootest2,2,function(x) quantile(x,0.025)); mu2.high = apply(
    bootest2,2,function(x) quantile(x,0.975));

# calculate Stage 2 Summary Statistics
mu2 = dataset$mu2
POA = mean(sign(mu2.hat[1:n]-mu2.hat[n+1:n])==sign(mu2[1:n]-mu2[n+1:n]) |
    mu2[1:n]==mu2[n+1:n])
```

```
bias = mean(mu2.hat-mu2)
RMSE = sqrt(mean((mu2.hat-mu2)**2))
w95 = mean(mu2.high-mu2.low)
c95 = mean(mu2.low < mu2 & mu2 < mu2.high)
stats2 = c(POA,bias,RMSE,w95,c95)

# m-out-of-n percentile bootstrap for confidence intervals on mu1
Z2 <- diag(y2-as.vector(X2%*%beta2.hat))%*%X2%*%(n*X2tX2.inv)/sqrt(n-q2);
    Cov2 <- t(Z2)%*%Z2
p <- mean(abs(cbind(a2,a2*o1,a2*a1,a2*y1)%*%beta2.hat[6:9])/sqrt(diag(cbind
    (1,o1,a1,y1)%*%Cov2[6:9,6:9]%*%rbind(1,o1,a1,y1))/n) <= qnorm(0.9995))
m <- ceiling(n^(1-p*(fixedXi/(1+fixedXi))))
bootest1 <- matrix(NA,nboot,2*n); g = 0
while(g<nboot){
  index <- sample(n,m,replace = TRUE); beta2.hat.temp <- as.vector(coef(lm(
      y2[index]~X2[index,]-1)))
  if(sum(is.na(beta2.hat.temp))==0){
    y.tilde.temp <- y1+cbind(1,o1,a1,a1*o1,y1)%*%beta2.hat.temp[1:5]+abs(
        cbind(1,o1,a1,y1)%*%beta2.hat.temp[6:9])
    beta1.hat.temp <- as.vector(coef(lm(y.tilde.temp[index]~X1[index,]-1)))
    if(sum(is.na(beta1.hat.temp))==0){
      g <- g+1; bootest1[g,] <- 2*mu1.hat-as.vector(X1.pred%*%cbind(beta1.
          hat.temp))
    }
  }
}
mu1.low = apply(bootest1,2,function(x) quantile(x,0.025)); mu1.high = apply(
    bootest1,2,function(x) quantile(x,0.975));

# calculate Stage 1 Summary Statistics
mu1 = dataset$mu1
POA = mean(sign(mu1.hat[1:n]-mu1.hat[n+1:n])==sign(mu1[1:n]-mu1[n+1:n]) |
```

```
    mu1[1:n]==mu1[n+1:n])
  bias = mean(mu1.hat-mu1)
  RMSE = sqrt(mean((mu1.hat-mu1)**2))
  w95 = mean(mu1.high-mu1.low)
  c95 = mean(mu1.low < mu1 & mu1 < mu1.high)
  stats1 = c(POA,bias,RMSE,w95,c95)

  # Combine Stats
  stats = round(c(stats1,stats2),4)

  return(stats)
}
```

```
### Q-Learning using Additive Regression Models with the n-out-of-n bootstrap
library(mgcv)
ql.gam = function(dataset,nboot=2000){
  # data
  n = nrow(dataset$data); o1 = dataset$data[,"o1"]; a1 = dataset$data[,"a1"]
  y1 = dataset$data[,"y1"]; a2 = dataset$data[,"a2"]; y2 = dataset$data[,"y2"]

  # implement method
  X2.pred = rbind(cbind(o1,a1,y1,1),cbind(o1,a1,y1,-1)); colnames(X2.pred) = c
      ("o1","a1","y1","a2")
  mu2.hat = as.vector(predict(gam(y2~factor(a1)+s(o1,by=factor(a1))+factor(a2)
      +s(o1,by=factor(a2))+a2*a1+s(y1,by=factor(a2))),newdata=data.frame(cbind
      (y2=NA,X2.pred))))
  y.tilde = y1+pmax(mu2.hat[1:n],mu2.hat[n+1:n]); X1.pred = rbind(cbind(o1,1),
      cbind(o1,-1)); colnames(X1.pred) = c("o1","a1")
  mu1.hat = as.vector(predict(gam(y.tilde~factor(a1)+s(o1,by=factor(a1))),
```

```
    newdata=data.frame(cbind(y.tilde=NA,X1.pred))))

### n-out-of-n percential bootstrap for confidence intervals on mu2 and mu1
bootest1 = bootest2 = matrix(NA,nboot,2*n)
for(g in 1:nboot){
  index <- sample(n,n,replace=TRUE); X2.pred.temp = rbind(cbind(o1,a1,y1,1)[
      index,],cbind(o1,a1,y1,-1)[index,]); colnames(X2.pred.temp) = c("o1","
      a1","y1","a2")
  bootest2[g,] = 2*mu2.hat-as.vector(predict(gam(y2~factor(a1)+s(o1,by=
      factor(a1))+factor(a2)+s(o1,by=factor(a2))+a2*a1+s(y1,by=factor(a2)),
      data=data.frame(cbind(o1,a1,y1,a2,y2)[index,])),newdata=data.frame(
      cbind(y2=NA,X2.pred))))
  mu2.hat.temp = as.vector(predict(gam(y2~factor(a1)+s(o1,by=factor(a1))+
      factor(a2)+s(o1,by=factor(a2))+a2*a1+s(y1,by=factor(a2)),data=data.
      frame(cbind(o1,a1,y1,a2,y2)[index,])),newdata=data.frame(cbind(y2=NA,
      X2.pred.temp))))
  y.tilde.temp = y1[index]+pmax(mu2.hat.temp[1:n],mu2.hat.temp[n+1:n])
  bootest1[g,] = 2*mu1.hat-as.vector(predict(gam(y.tilde~factor(a1)+s(o1,by=
      factor(a1)),data=data.frame(y.tilde=y.tilde.temp,o1=o1[index],a1=a1[
      index])),newdata=data.frame(cbind(y.tilde=NA,X1.pred))))
}
mu2.low = apply(bootest2,2,function(x) quantile(x,0.025)); mu2.high = apply(
    bootest2,2,function(x) quantile(x,0.975));
mu1.low = apply(bootest1,2,function(x) quantile(x,0.025)); mu1.high = apply(
    bootest1,2,function(x) quantile(x,0.975));

# calculate Stage 2 Summary Statistics
mu2 = dataset$mu2
POA = mean(sign(mu2.hat[1:n]-mu2.hat[n+1:n])==sign(mu2[1:n]-mu2[n+1:n]) |
    mu2[1:n]==mu2[n+1:n])
bias = mean(mu2.hat-mu2)
RMSE = sqrt(mean((mu2.hat-mu2)**2))
```

```
  w95 = mean(mu2.high-mu2.low)
  c95 = mean(mu2.low < mu2 & mu2 < mu2.high)
  stats2 = c(POA, bias ,RMSE, w95, c95)

  # calculate Stage 1 Summary Statistics
  mu1 = dataset$mu1
  POA = mean(sign(mu1.hat[1:n]-mu1.hat[n+1:n])==sign(mu1[1:n]-mu1[n+1:n]) |
      mu1[1:n]==mu1[n+1:n])
  bias = mean(mu1.hat-mu1)
  RMSE = sqrt(mean((mu1.hat-mu1)**2))
  w95 = mean(mu1.high-mu1.low)
  c95 = mean(mu1.low < mu1 & mu1 < mu1.high)
  stats1 = c(POA, bias ,RMSE, w95, c95)

  # Combine Stats
  stats = round(c(stats1 ,stats2),4)

  return(stats)
}
```

simulation2-real-linear-associations.R

```
#
    ##############################################################################

### Program to Reproduce the Results for Scenario 1 of our Second Simulation
    Study ###
#
    ##############################################################################

source('simulation2-real-functions.R')

### Function to Generate Datasets
get.dataset = function(n){
  # Sample Parameters
  phi1 = cbind(rnorm(4,0,c(rep(0.4,3),0.2)))
  phi2 = cbind(rnorm(9,0,c(0.4,0.4,0.4,0.2,0.4,0.4,0.2,0.2,0.2)))

  # baseline covariate
  o1 = 2*(rbeta(n,1,1)-0.5)

  # treatment 1
  a1 = 2*rbinom(n,1,0.5)-1

  # stage 1 payoff
  y1 = rnorm(n,cbind(1,o1,a1,a1*o1)%*%phi1,1)

  # treatment 2
  a2 = 2*rbinom(n,1,0.5)-1

  # stage 2 payoff
  y2 = rnorm(n,cbind(1,o1,a1,a1*o1,y1,a2,a2*o1,a2*a1,a2*y1)%*%phi2,1)
```

```
  # conditional mean payoff functions for each treatment option at each stage
  mu2.p = cbind(cbind(1,o1,a1,a1*o1,y1),cbind(1,o1,a1,y1))%*%phi2
  mu2.n = cbind(cbind(1,o1,a1,a1*o1,y1),-cbind(1,o1,a1,y1))%*%phi2
  mu1.p = sapply(1:n,function(i){
    y1.p = rnorm(100000,cbind(1,o1[i],1,o1[i])%*%phi1,1)
    a2.opt.p = sign(cbind(1,o1[i],1,y1.p)%*%phi2[6:9])
    mu1.p = mean(y1.p + cbind(1,o1[i],1,o1[i],y1.p,a2.opt.p,a2.opt.p*o1[i],a2.
        opt.p,a2.opt.p*y1.p)%*%phi2)
    return(mu1.p)})
  mu1.n = sapply(1:n,function(i){
    y1.n = rnorm(100000,cbind(1,o1[i],-1,-o1[i])%*%phi1,1)
    a2.opt.n = sign(cbind(1,o1[i],-1,y1.n)%*%phi2[6:9])
    mu1.n = mean(y1.n + cbind(1,o1[i],-1,-o1[i],y1.n,a2.opt.n,a2.opt.n*o1[i],-
        a2.opt.n,a2.opt.n*y1.n)%*%phi2)
    return(mu1.n)})
  mu2 = c(mu2.p,mu2.n); mu1 = c(mu1.p,mu1.n)

  return(list(data=cbind(o1,a1,y1,a2,y2), mu1=mu1, mu2=mu2))
}

### Generate <niters> datasets each with <n> observations for case <case>
set.seed(1985); niters = 1000; n = 300
datasets <- lapply(1:niters,function(iter) get.dataset(n=n))

### Fit Each Method to Each Dataset
stats.bml.glm <- t(sapply(1:niters,function(iter) bml.glm(dataset=datasets[[
    iter]])))
stats.ql.glm <- t(sapply(1:niters,function(iter) ql.glm(dataset=datasets[[iter
    ]])))
stats.bml.bart <- t(sapply(1:niters,function(iter) bml.bart(dataset=datasets[[
    iter]])))
stats.ql.gam <- t(sapply(1:niters,function(iter) ql.gam(dataset=datasets[[iter
```

```
]]))))

### Compile Results
library(xtable)
res <- rbind(colMeans(stats.bml.glm),colMeans(stats.ql.glm),colMeans(stats.bml
    .bart),colMeans(stats.ql.gam))
print(xtable(res,digits=3))
```

simulation2-real-nonlinear-associations.R

```
#
   ##############################################################################

### Program to Reproduce the Results for Scenario 2 of our Second Simulation
    Study ###
#
   ##############################################################################

source('simulation2-real-functions.R')

### Function to Generate Datasets
get.dataset = function(n){

  # baseline covariate
  o1 = 2*(rbeta(n,1,1)-0.5)

  # treatment 1
  a1 = 2*rbinom(n,1,0.5)-1

  # stage 1 payoff
  eta1 = a1*(2/(1+exp(-6*o1))-1)
  y1 = rnorm(n,eta1,0.25)

  # treatment 2
  a2 = 2*rbinom(n,1,0.5)-1

  # stage 2 payoff
  eta2 = a2*(cos(pi/2*y1)-0.25)
  y2 = rnorm(n,eta2,0.25)

  # conditional mean payoff functions for each treatment option at each stage
```

```
  mu2.p =  (cos(pi/2*y1)-0.25)
  mu2.n = -(cos(pi/2*y1)-0.25)
  mu1.p = sapply(1:n,function(i){
    y1.p = rnorm(100000,(2/(1+exp(-6*o1[i]))-1),0.25)
    mu1.p = mean(y1.p+abs(cos(pi/2*y1.p)-0.25))
    return(mu1.p)})
  mu1.n = sapply(1:n,function(i){
    y1.n = rnorm(100000,-(2/(1+exp(-6*o1[i]))-1),0.25)
    mu1.n = mean(y1.n+abs(cos(pi/2*y1.n)-0.25))
    return(mu1.n)})
  mu2 = c(mu2.p,mu2.n); mu1 = c(mu1.p,mu1.n)

  return(list(data=cbind(o1,a1,y1,a2,y2), mu1=mu1, mu2=mu2))
}

### Generate <niters> datasets each with <n> observations for case <case>
set.seed(1985); niters = 1000; n = 300
datasets <- lapply(1:niters,function(iter) get.dataset(n=n))

### Fit Each Method to Each Dataset
stats.bml.glm <- t(sapply(1:niters,function(iter) bml.glm(dataset=datasets[[
    iter]])))
stats.ql.glm <- t(sapply(1:niters,function(iter) ql.glm(dataset=datasets[[iter
    ]])))
stats.bml.bart <- t(sapply(1:niters,function(iter) bml.bart(dataset=datasets[[
    iter]])))
stats.ql.gam <- t(sapply(1:niters,function(iter) ql.gam(dataset=datasets[[iter
    ]])))

### Compile Results
library(xtable)
res <- rbind(colMeans(stats.bml.glm),colMeans(stats.ql.glm),colMeans(stats.bml
```

```
    .bart),colMeans(stats.ql.gam))
print(xtable(res,digits=3))
```

simulation2-binary-functions.R

```
#Bayesian Machine Learning using logistic regression models with improper
    priors
library(BayesLogit)
bml.glm = function(dataset,nsamps=2000,warmup=100){
  # data
  n1 = nrow(dataset$data); o1 = dataset$data[,"o1"]; a1 = dataset$data[,"a1"];
       y1 = dataset$data[,"y1"]
  n2 = sum(y1==0); o2 = dataset$data[,"o2"]; a2 = dataset$data[,"a2"]; y2 =
      dataset$data[,"y2"]

  # construct design matrices for stage 2 and stage 1 models
  X2 = cbind(1,o1,a1,a1*o1,o2,a2,a2*o1,a2*a1,a2*o2); X1 = cbind(1,o1,a1,a1*o1)
      ; q2 = ncol(X2); q1 = ncol(X1)
  X2.pred = rbind(cbind(cbind(1,o1,a1,a1*o1,o2),cbind(1,o1,a1,o2))[y1==0,],
      cbind(cbind(1,o1,a1,a1*o1,o2),-cbind(1,o1,a1,o2))[y1==0,])
  X1.pred = rbind(cbind(cbind(1,o1),cbind(1,o1)),cbind(cbind(1,o1),-cbind(1,o1
      )))

  # Sample the stage 2 parameter from its posterior distribution
  capture.output(beta2.samps <- logit(y2[y1==0],X2[y1==0,],samp=nsamps,burn=
      warmup)$beta,file="NUL")
  #capture.output(beta2.samps <- logit(y2[y1==0],X2[y1==0,],samp=nsamps,burn=
      warmup)$beta,file="/dev/null")

  # Calculate Stage 2 Summary Statistics
  mu2 = dataset$mu2
  mu2.samps = 1/(1+exp(-X2.pred%*%t(beta2.samps)))
  mu2.hat = rowMeans(mu2.samps)
  mu2.low = apply(mu2.samps,1,function(x) quantile(x,0.025))
  mu2.high = apply(mu2.samps,1,function(x) quantile(x,0.975))
  POA = mean(sign(mu2.hat[1:n2]-mu2.hat[n2+1:n2])==sign(mu2[1:n2]-mu2[n2+1:n2
```

```
    ]))
bias = mean(mu2.hat-mu2)
RMSE = sqrt(mean((mu2.hat-mu2)**2))
w95 = mean(mu2.high-mu2.low)
c95 = mean(mu2.low < mu2 & mu2 < mu2.high)
stats2 = c(POA,bias,RMSE,w95,c95)

# Sample the stage 1 parameter from its poseterior distribution using the
    BIG sampler
beta1.samps = matrix(NA,nsamps,q1); d2.opt.samps = sample(nsamps)
for(samp in 1:nsamps){
  # Step 1. Sample the optimal stage 2 decision rule and determine the
      optimal stage 2 treatment for each patient
  a2.opt = sign(cbind(1,o1,a1,o2)[y1==0,]%*%beta2.samps[d2.opt.samps[samp
      ],6:9])
  X2.opt = cbind(cbind(1,o1,a1,a1*o1,o2)[y1==0,],apply(cbind(1,o1,a1,o2)[y1
      ==0,],2,function(x) a2.opt*x))

  # Step 2. Sample missing payoffs under the sampled optimal stage 2
      decision rule
  y2.opt = y1; y2.opt[y1==0] = ifelse(a2[y1==0]==a2.opt,y2[y1==0],rbinom(n2
      ,1,1/(1+exp(-X2.opt%*%beta2.samps[samp,]))))

  # Step 3. Sample parameter from the full conditional stage 1 posterior
      distribution
  capture.output(beta1.samps[samp,] <- logit(y2.opt,X1,samp=1,burn=warmup)$
      beta[1,],file="NUL")
  #capture.output(beta1.samps[samp,] <- logit(y2.opt,X1,samp=1,burn=warmup)$
      beta[1,],file="/dev/null")
}

# Calculate Stage 1 Summary Statistics
```

```
  mu1 = dataset$mu1
  mu1.samps = 1/(1+exp(-X1.pred%*%t(beta1.samps)))
  mu1.hat = rowMeans(mu1.samps)
  mu1.low = apply(mu1.samps,1,function(x) quantile(x,0.025))
  mu1.high = apply(mu1.samps,1,function(x) quantile(x,0.975))
  POA = mean(sign(mu1.hat[1:n1]-mu1.hat[n1+1:n1])==sign(mu1[1:n1]-mu1[n1+1:n1
      ]))
  bias = mean(mu1.hat-mu1)
  RMSE = sqrt(mean((mu1.hat-mu1)**2))
  w95 = mean(mu1.high-mu1.low)
  c95 = mean(mu1.low < mu1 & mu1 < mu1.high)
  stats1 = c(POA, bias ,RMSE, w95, c95)

  # Combine Stage 2 and Stage 1 Summary Statistics
  stats = round(c(stats1,stats2),4)

  return(stats)
}



#Bayesian Machine Learning with probit BART models
library(BayesTree)
bml.bart = function(dataset,nsamps=2000,warmup=100,ntree=20,prior=0.95){
  # data
  n1 = nrow(dataset$data); o1 = dataset$data[,"o1"]; a1 = dataset$data[,"a1"];
       y1 = dataset$data[,"y1"]
  n2 = sum(y1==0); o2 = dataset$data[,"o2"]; a2 = dataset$data[,"a2"]; y2 =
      dataset$data[,"y2"]

  # Sample the stage 2 parameter from its posterior distribution
  X2 = cbind(o1,a1,o2,a2); X2.pred = rbind(cbind(o1,a1,o2,1)[y1==0,],cbind(o1,
```

```
    a1,o2,-1)[y1==0,]) #Predictions
capture.output(mu2.samps <- pnorm(bart(X2[y1==0,],y2[y1==0],X2.pred,ntree=
    ntree,k=qnorm((prior+1)/2),binaryOffset=qnorm((sum(y2[y1==0])+1)/(n2+2))
    ,ndpost=nsamps,nskip=warmup,keeptrainfits=FALSE,verbose=FALSE)$yhat.test
    ),file="NUL")

# Calculate Stage 2 Summary Statistics
mu2 = dataset$mu2
mu2.hat = colMeans(mu2.samps)
mu2.low = apply(mu2.samps,2,function(x) quantile(x,0.025))
mu2.high = apply(mu2.samps,2,function(x) quantile(x,0.975))
POA = mean(sign(mu2.hat[1:n2]-mu2.hat[n2+1:n2])==sign(mu2[1:n2]-mu2[n2+1:n2
    ]))
bias = mean(mu2.hat-mu2)
RMSE = sqrt(mean((mu2.hat-mu2)**2))
w95 = mean(mu2.high-mu2.low)
c95 = mean(mu2.low < mu2 & mu2 < mu2.high)
stats2 = c(POA,bias,RMSE,w95,c95)

# Sample the stage 1 parameter from its poseterior distribution using the
    BIG sampler
X1 = cbind(o1,a1); X1.pred = rbind(cbind(o1,1),cbind(o1,-1)); mu1.samps =
    matrix(NA,nrow=nsamps,ncol=2*n1); d2.opt.samps = sample(nsamps)
for(samp in 1:nsamps){
  # Step 1. Sample the optimal stage 2 rule and determine optimal stage 2
      treatment for each patient under this rule
  a2.opt = sign(mu2.samps[d2.opt.samps[samp],1:n2]-mu2.samps[d2.opt.samps[
      samp],n2+1:n2])

  # Step 2. Sample missing payoffs under the sampled optimal stage 2
      decision rule
  y2.opt = y1; y2.opt[y1==0] = ifelse(a2[y1==0]==a2.opt,y2[y1==0],rbinom(n2
```

```
      ,1,(a2.opt==1)*mu2.samps[samp,1:n2]+(a2.opt==-1)*mu2.samps[samp,n2+1:
      n2]))

    # Step 3. Sample parameters from the full conditional stage 1 posterior
      distribution
    capture.output(mu1.samps[samp,] <- pnorm(bart(X1,y2.opt,X1.pred,ntree=
      ntree,k=qnorm((prior+1)/2),binaryOffset=qnorm((sum(y2[y1==0])+sum(y1)
      +1)/(n1+2)),ndpost=2,nskip=warmup,keeptrainfits=FALSE,verbose=FALSE)$
      yhat.test[2,]),file="NUL")
  }

  # Calculate Stage 1 Summary Statistics
  mu1 = dataset$mu1
  mu1.hat = colMeans(mu1.samps)
  mu1.low = apply(mu1.samps,2,function(x) quantile(x,0.025))
  mu1.high = apply(mu1.samps,2,function(x) quantile(x,0.975))
  POA = mean(sign(mu1.hat[1:n1]-mu1.hat[n1+1:n1])==sign(mu1[1:n1]-mu1[n1+1:n1
    ]))
  bias = mean(mu1.hat-mu1)
  RMSE = sqrt(mean((mu1.hat-mu1)**2))
  w95 = mean(mu1.high-mu1.low)
  c95 = mean(mu1.low < mu1 & mu1 < mu1.high)
  stats1 = c(POA,bias,RMSE,w95,c95)

  # Combine Stage 2 and Stage 1 Summary Statistics
  stats = round(c(stats1,stats2),4)

  return(stats)
}
```

```
### Q-learning with generalized linear models
ql.glm = function(dataset,nboot=2000){
  # data
  n1 = nrow(dataset$data); o1 = dataset$data[,"o1"]; a1 = dataset$data[,"a1"];
       y1 = dataset$data[,"y1"]
  n2 = sum(y1==0); o2 = dataset$data[,"o2"]; a2 = dataset$data[,"a2"]; y2 =
      dataset$data[,"y2"]

  # construct design matrices for stage 2 and stage 1 models
  X2 = cbind(1,o1,a1,a1*o1,o2,a2,a2*o1,a2*a1,a2*o2); X1 = cbind(1,o1,a1,a1*o1)
  X2.pred = rbind(cbind(cbind(1,o1,a1,a1*o1,o2),cbind(1,o1,a1,o2))[y1==0,],
      cbind(cbind(1,o1,a1,a1*o1,o2),-cbind(1,o1,a1,o2))[y1==0,])
  X1.pred = rbind(cbind(cbind(1,o1),cbind(1,o1)),cbind(cbind(1,o1),-cbind(1,o1
      )))

  # implement method
  beta2.hat = as.vector(glm(y2[y1==0]~X2[y1==0,]-1,family="quasibinomial")$
      coeff)
  mu2.hat = 1/(1+exp(-X2.pred%*%beta2.hat))
  a2.opt = sign(cbind(1,o1,a1,o2)[y1==0,]%*%beta2.hat[6:9])
  X2.opt = cbind(cbind(1,o1,a1,a1*o1,o2)[y1==0,],apply(cbind(1,o1,a1,o2)[y1
      ==0,],2,function(x) a2.opt*x))
  y.tilde = y1; y.tilde[y1==0] = 1/(1+exp(-X2.opt%*%beta2.hat))
  beta1.hat = as.vector(glm(y.tilde~X1-1,family="quasibinomial")$coeff)
  mu1.hat = 1/(1+exp(-X1.pred%*%beta1.hat))

  # n-out-of-n bootstrap
  bootest2 <- matrix(NA,nboot,2*n2); bootest1 <- matrix(NA,nboot,2*n1)
  for(g in 1:nboot){
    index <- sample(n1,n1,replace=TRUE)
    X2.temp = X2[index,]; y2.temp = y2[index]; X1.temp = X1[index,]; y1.temp =
        y1[index]
```

```
    beta2.hat.temp = as.vector(glm(y2.temp[y1.temp==0]~X2.temp[y1.temp==0,]-1,
        family="quasibinomial")$coeff)
    bootest2[g,] = 2*mu2.hat - 1/(1+exp(-X2.pred%*%beta2.hat.temp))
    a2.opt.temp = sign(cbind(1,o1,a1,o2)[index,][y1.temp==0,]%*%beta2.hat.temp
        [6:9])
    X2.opt.temp = cbind(cbind(1,o1,a1,a1*o1,o2)[index,][y1.temp==0,],apply(
        cbind(1,o1,a1,o2)[index,][y1.temp==0,],2,function(x) a2.opt.temp*x))
    y.tilde.temp = y1.temp; y.tilde.temp[y1.temp==0] = 1/(1+exp(-X2.opt.temp%*
        %beta2.hat.temp))
    beta1.hat.temp = as.vector(glm(y.tilde.temp~X1.temp-1,family="
        quasibinomial")$coeff)
    bootest1[g,] = 2*mu1.hat - 1/(1+exp(-X1.pred%*%beta1.hat.temp))
}
mu2.low = apply(bootest2,2,function(x) quantile(x,0.025)); mu2.high = apply(
    bootest2,2,function(x) quantile(x,0.975));
mu1.low = apply(bootest1,2,function(x) quantile(x,0.025)); mu1.high = apply(
    bootest1,2,function(x) quantile(x,0.975));

# Calculate Stage 2 Summary Statistics
mu2 = dataset$mu2
POA = mean(sign(mu2.hat[1:n2]-mu2.hat[n2+1:n2])==sign(mu2[1:n2]-mu2[n2+1:n2
    ]))
bias = mean(mu2.hat-mu2)
RMSE = sqrt(mean((mu2.hat-mu2)**2))
w95 = mean(mu2.high-mu2.low)
c95 = mean(mu2.low < mu2 & mu2 < mu2.high)
stats2 = c(POA,bias,RMSE,w95,c95)

# Calculate Stage 1 Summary Statistics
mu1 = dataset$mu1
POA = mean(sign(mu1.hat[1:n1]-mu1.hat[n1+1:n1])==sign(mu1[1:n1]-mu1[n1+1:n1
    ]))
```

```
  bias = mean(mu1.hat-mu1)
  RMSE = sqrt(mean((mu1.hat-mu1)**2))
  w95 = mean(mu1.high-mu1.low)
  c95 = mean(mu1.low < mu1 & mu1 < mu1.high)
  stats1 = c(POA,bias,RMSE,w95,c95)

  # Combine Stats
  stats = round(c(stats1,stats2),4)

  return(stats)
}



### Q-learning with generalized additive models
library(mgcv)
ql.gam = function(dataset,nboot=2000){
  # data
  n1 = nrow(dataset$data); o1 = dataset$data[,"o1"]; a1 = dataset$data[,"a1"];
       y1 = dataset$data[,"y1"]
  n2 = sum(y1==0); o2 = dataset$data[,"o2"]; a2 = dataset$data[,"a2"]; y2 =
      dataset$data[,"y2"]

  # prediction matricies
  X2.pred = rbind(cbind(o1,a1,o2,1)[y1==0,],cbind(o1,a1,o2,-1)[y1==0,]);
      colnames(X2.pred) = c("o1","a1","o2","a2")
  X1.pred = rbind(cbind(o1,1),cbind(o1,-1)); colnames(X1.pred) = c("o1","a1")

  # implement method
  mu2.hat = 1/(1+exp(-as.vector(predict(gam(y2~factor(a1)+s(o1,by=factor(a1))+
      factor(a2)+s(o1,by=factor(a2))+a2*a1+s(o2,by=factor(a2)),data=data.frame
      (cbind(o1,a1,y1,o2,a2,y2)),subset=c(y1==0),family="quasibinomial"),
```

```
    newdata=data.frame(cbind(y2=NA,X2.pred))))))
y.tilde = y1; y.tilde[y1==0] = pmax(mu2.hat[1:n2],mu2.hat[n2+1:n2])
mu1.hat = 1/(1+exp(-as.vector(predict(gam(y.tilde~factor(a1)+s(o1,by=factor(
    a1)),data=data.frame(cbind(o1,a1,y1,o2,a2,y2)),family="quasibinomial"),
    newdata=data.frame(cbind(y.tilde=NA,X1.pred))))))


# n-out-of-n bootstrap (This doesn't run, gam() is unstable with replicated
    observations, see http://stats.stackexchange.com/questions/190348/can-i-
    use-bootstrapping-to-estimate-the-uncertainty-in-a-maximum-value-of-a-
    gam)
#bootest2 <- matrix(NA,nboot,2*n2); bootest1 <- matrix(NA,nboot,2*n1)
#for(g in 1:nboot){
#  index <- sample(n1,n1,replace=TRUE); n2.temp = sum(y1[index]==0); X2.pred
    .temp = rbind(cbind(o1,a1,o2,1)[index,][y1[index]==0,],cbind(o1,a1,o2
    ,-1)[index,][y1[index]==0,]); colnames(X2.pred.temp) = c("o1","a1","o2
    ","a2")
#  bootest2[g,] = 2*mu2.hat-1/(1+exp(-as.vector(predict(gam(y2~factor(a1)+s(
    o1,by=factor(a1))+factor(a2)+s(o1,by=factor(a2))+a2*a1+s(o2,by=factor(a2
    ))),data=data.frame(cbind(o1,a1,o2,a2,y2)[index,][y1[index]==0,]),family
    ="binomial"),newdata=data.frame(cbind(y2=NA,X2.pred))))))
#  mu2.hat.temp = 1/(1+exp(-as.vector(predict(gam(y2~factor(a1)+s(o1,by=
    factor(a1))+factor(a2)+s(o1,by=factor(a2))+a2*a1+s(o2,by=factor(a2)),
    data=data.frame(cbind(o1,a1,o2,a2,y2)[index,][y1[index]==0,]),family="
    binomial"),newdata=data.frame(cbind(y2=NA,X2.pred.temp))))))
#  y.tilde.temp = y1[index]; y.tilde.temp[y1[index]==0] = pmax(mu2.hat.temp
    [1:n2.temp],mu2.hat.temp[n2.temp+1:n2.temp])
#  bootest1[g,] = 2*mu1.hat-1/(1+exp(-as.vector(predict(gam(y.tilde~factor(
    a1)+s(o1,by=factor(a1)),data=data.frame(y.tilde=y.tilde.temp,a1=a1[index
    ],o1=o1[index]),family="quasibinomial"),newdata=data.frame(y.tilde=NA,X1
    .pred)))))
#}
#mu2.low = apply(bootest2,2,function(x) quantile(x,0.025)); mu2.high = apply
```

```
    (bootest2,2,function(x) quantile(x,0.975));
  #mu1.low = apply(bootest1,2,function(x) quantile(x,0.025)); mu1.high = apply
    (bootest1,2,function(x) quantile(x,0.975));

  # Calculate Stage 2 Summary Statistics
  mu2 = dataset$mu2
  POA = mean(sign(mu2.hat[1:n2]-mu2.hat[n2+1:n2])==sign(mu2[1:n2]-mu2[n2+1:n2
    ]))
  bias = mean(mu2.hat-mu2)
  RMSE = sqrt(mean((mu2.hat-mu2)**2))
  w95 = NA #w95 = mean(mu2.high-mu2.low)
  c95 = NA #c95 = mean(mu2.low < mu2 & mu2 < mu2.high)
  stats2 = c(POA,bias,RMSE,w95,c95)

  # Calculate Stage 1 Summary Statistics
  mu1 = dataset$mu1
  POA = mean(sign(mu1.hat[1:n1]-mu1.hat[n1+1:n1])==sign(mu1[1:n1]-mu1[n1+1:n1
    ]))
  bias = mean(mu1.hat-mu1)
  RMSE = sqrt(mean((mu1.hat-mu1)**2))
  w95 = NA #w95 = mean(mu1.high-mu1.low)
  c95 = NA #c95 = mean(mu1.low < mu1 & mu1 < mu1.high)
  stats1 = c(POA,bias,RMSE,w95,c95)

  # Combine Stats
  stats = round(c(stats1,stats2),4)

  return(stats)
}
```

simulation2-binary-linear-associations.R

```
#
   ##############################################################################

### Program to Reproduce the Results for Scenario 3 of our Second Simulation
    Study ###
#
   ##############################################################################

source('simulation2-binary-functions.R')

# generate data
get.dataset = function(n){

  # Sample Parameters
  phi1 = cbind(rnorm(4,0,c(rep(0.6,3),0.4)))
  gamma = cbind(rnorm(4,0,c(rep(0.6,3),0.4)))
  phi2 = cbind(rnorm(9,0,c(0.6,0.6,0.6,0.4,0.6,0.6,0.4,0.4,0.4)))

  # baseline covariate
  o1 = 2*(rbeta(n,1,1)-0.5)

  # treatment 1
  a1 = 2*rbinom(n,1,0.5)-1

  # updated covariate
  eta1 = cbind(1,o1,a1,a1*o1)%*%phi1
  y1 = rbinom(n,1,1/(1+exp(-eta1)))
  o2 = 2*(rbeta(n,5/(1+exp(-cbind(1,o1,a1,a1*o1)%*%gamma)),5/(1+exp(cbind(1,o1
      ,a1,a1*o1)%*%gamma)))-0.5)

  # treatment 2
```

```
a2 = 2*rbinom(n,1,0.5)-1

# final payoff
eta2 = cbind(1,o1,a1,a1*o1,o2,a2,a2*o1,a2*a1,a2*o2)%*%phi2
y2 = rbinom(n,1,1/(1+exp(-eta2)))
y2[y1==1] = o2[y1==1] = a2[y1==1] = NA

# conditional Mean Payoff Functions for Each Treatment Option at Each Stage
mu2.p = 1/(1+exp(-cbind(cbind(1,o1,a1,a1*o1,o2)[y1==0,],cbind(1,o1,a1,o2)[y1
    ==0,])%*%phi2))
mu2.n = 1/(1+exp(-cbind(cbind(1,o1,a1,a1*o1,o2)[y1==0,],-cbind(1,o1,a1,o2)[
    y1==0,])%*%phi2))
mu1.p = sapply(1:n,function(i){
  pi1.p = as.vector(1/(1+exp(-cbind(1,o1[i],1,o1[i])%*%phi1)))
  o2.p = 2*(rbeta(100000,5/(1+exp(-cbind(1,o1[i],1,o1[i])%*%gamma)),5/(1+exp
      (cbind(1,o1[i],1,o1[i])%*%gamma)))-0.5)
  a2.opt.p = sign(cbind(1,o1[i],1,o2.p)%*%phi2[6:9])
  pi2.p = mean(1/(1+exp(-cbind(1,o1[i],1,o1[i],o2.p,a2.opt.p,a2.opt.p*o1[i],
      a2.opt.p,a2.opt.p*o2.p)%*%phi2)))
  mu1.p = pi1.p+(1-pi1.p)*pi2.p
  return(mu1.p)})
mu1.n = sapply(1:n,function(i){
  pi1.n = as.vector(1/(1+exp(-cbind(1,o1[i],-1,-o1[i])%*%phi1)))
  o2.n = 2*(rbeta(100000,5/(1+exp(-cbind(1,o1[i],-1,-o1[i])%*%gamma)),5/(1+
      exp(cbind(1,o1[i],-1,-o1[i])%*%gamma)))-0.5)
  a2.opt.n = sign(cbind(1,o1[i],-1,o2.n)%*%phi2[6:9])
  pi2.n = mean(1/(1+exp(-cbind(1,o1[i],-1,-o1[i],o2.n,a2.opt.n,a2.opt.n*o1[i
      ],-a2.opt.n,a2.opt.n*o2.n)%*%phi2)))
  mu1.n = pi1.n+(1-pi1.n)*pi2.n
  return(mu1.n)})
mu2 = c(mu2.p,mu2.n); mu1 = c(mu1.p,mu1.n)
```

```
  return(list(data=cbind(o1,a1,y1,o2,a2,y2), mu1=mu1, mu2=mu2))
}


### Generate <niters> datasets each with <n> observations for case <case>
set.seed(1985); niters = 1000; n = 300
datasets <- lapply(1:niters,function(iter) get.dataset(n=n))

### Fit Each Method to Each Dataset
stats.bml.glm <- t(sapply(1:niters,function(iter) bml.glm(dataset=datasets[[
    iter]])))
stats.ql.glm <- t(sapply(1:niters,function(iter) ql.glm(dataset=datasets[[iter
    ]])))
stats.bml.bart <- t(sapply(1:niters,function(iter) bml.bart(dataset=datasets[[
    iter]])))
stats.ql.gam <- t(sapply(1:niters,function(iter) ql.gam(dataset=datasets[[iter
    ]])))

### Compile Results
library(xtable)
res <- rbind(colMeans(stats.bml.glm),colMeans(stats.ql.glm),colMeans(stats.bml
    .bart),colMeans(stats.ql.gam))
print(xtable(res,digits=3))
```

simulation2-binary-nonlinear-associations.R

```
#
   ##############################################################################

### Program to Reproduce the Results for Scenario 4 of our Second Simulation
    Study ###
#
   ##############################################################################

source('simulation2-binary-functions.R')

#Function to Generate Datasets
get.dataset = function(n){

  # baseline covariate
  o1 = 2*(rbeta(n,1,1)-0.5)

  # treatment 1
  a1 = 2*rbinom(n,1,0.5)-1

  # interim payoff and covariate
  eta1 = a1*sin(pi/1.5*o1)-0.5
  y1 = rbinom(n,1,1/(1+exp(-eta1)))
  o2 = 2*(rbeta(n,2/(1+exp(-0.1*(o1+a1))),2/(1+exp(+0.1*(o1+a1))))-0.5)

  # treatment 2
  a2 = 2*rbinom(n,1,0.5)-1

  # stage 2 payoff
  eta2 = 3*a2*((1.5*o2)**2-0.25*(1.5*o2)**4-0.5)-1
  y2 = rbinom(n,1,1/(1+exp(-eta2)))
  y2[y1==1] = o2[y1==1] = a2[y1==1] = NA
```

```
  # conditional mean payoff functions for each treatment option at each stage
  mu2.p = 1/(1+exp(-(3*((1.5*o2[y1==0])**2-0.25*(1.5*o2[y1==0])**4-0.5)-1)))
  mu2.n = 1/(1+exp(-(-3*((1.5*o2[y1==0])**2-0.25*(1.5*o2[y1==0])**4-0.5)-1)))
  mu1.p = sapply(1:n,function(i){
    pi1.p = 1/(1+exp(-(sin(pi/1.5*o1[i])-0.5)))
    o2.p = 2*(rbeta(100000,2/(1+exp(-0.1*(o1[i]+1))),2/(1+exp(0.1*(o1[i]+1))))
        -0.5)
    pi2.p = mean(1/(1+exp(-(3*abs((1.5*o2.p)**2-0.25*(1.5*o2.p)**4-0.5)-1))))
    mu1.p = pi1.p+(1-pi1.p)*pi2.p
    return(mu1.p)})
  mu1.n = sapply(1:n,function(i){
    pi1.n = 1/(1+exp(-(-sin(pi/1.5*o1[i])-0.5)))
    o2.n = 2*(rbeta(100000,2/(1+exp(-0.1*(o1[i]-1))),2/(1+exp(0.1*(o1[i]-1))))
        -0.5)
    pi2.n = mean(1/(1+exp(-(3*abs((1.5*o2.n)**2-0.25*(1.5*o2.n)**4-0.5)-1))))
    mu1.n = pi1.n+(1-pi1.n)*pi2.n
    return(mu1.n)})
  mu2 = c(mu2.p,mu2.n); mu1 = c(mu1.p,mu1.n)

  return(list(data=cbind(o1,a1,y1,o2,a2,y2), mu1=mu1, mu2=mu2))
}

### Generate <niters> datasets each with <n> observations for case <case>
set.seed(1985); niters = 1000; n = 300
datasets <- lapply(1:niters,function(iter) get.dataset(n=n))

### Fit Each Method to Each Dataset
stats.bml.glm <- t(sapply(1:niters,function(iter) bml.glm(dataset=datasets[[
    iter]])))
stats.ql.glm <- t(sapply(1:niters,function(iter) ql.glm(dataset=datasets[[iter
    ]])))
```

```
stats.bml.bart <- t(sapply(1:niters,function(iter) bml.bart(dataset=datasets[[
    iter]])))
stats.ql.gam <- t(sapply(1:niters,function(iter) ql.gam(dataset=datasets[[iter
    ]])))

### Compile Results
library(xtable)
res <- rbind(colMeans(stats.bml.glm),colMeans(stats.ql.glm),colMeans(stats.bml
    .bart),colMeans(stats.ql.gam))
print(xtable(res,digits=3))
```