# Vignette for the package **bdpopt**

Sebastian Jobjörnsson

March 30, 2016

## Contents

# 1 Introduction

The R package described in this document has been implemented as a tool for studying certain types of decision problems that occur within the field of clinical trial optimisation. The work has been done within the framework of the EU project IDEAL (Integrated DEsign and AnaLysis of clinical trials in small population groups, see [1]). The project consists of several work packages focussed on different aspects of the statistical analysis of clinical trials for small population groups.

The objective of the R package is to provide a collection of functions that may be applied to the problem of optimising confirmatory clinical trials with respect to dose and sample size. Broadly, the functionality provided may be divided into two different levels. The core functions of the package may be used to solve Bayesian decision problems not necesserily associated with clinical trial decision making. On top of these core functions the package also provides a simplified interface to a few specific classes of clinical trial decision problems.

The Bayesian decision problems handled by **bdpopt** may be specified in terms of three components: a decision space, a probabilistic model and a utility function. The decision space will be denoted by $D$ and represents the set of possible decisions available to the decision maker, each of which is assumed to be encoded by a vector of numbers $d$. The goal of the decision maker when planning is find a decision $d^* \in D$ which is optimal in some sense, given the information available and a preference ranking for the different possible outcomes of a decision. Formally, it will be assumed that the available information consists of an observed value $y$ for a vector of numbers $Y$ and a specification of the conditional distribution (given $Y = y$ and a decision $d$) for a potential future observation $x$ of a vector of numbers $X$. This conditional distribution will be denoted by $\pi(x \mid y, d)$[1]. To complete the description of the decision problem, a decision maker also needs to specify the utility assigned to each possible decision-outcome pair $(X = x, d)$ given $Y = y$. This specification is encoded in terms of a utility function, which will be denoted by $u(x, y, d)$.

The major focus of the package is to provide support for solving Bayesian decision problems where the objective is to find the optimal decision $d^*$ given that $Y = y$ has been observed, i.e., to solve problems of the form

$$d^* = \arg\max_{d \in D} f(d), \text{ where}$$
$$f(d) = \mathbb{E}\left[u(X, Y, d) \mid Y = y, d\right] = \int_{\Omega_X} u(x, y, d)\pi(x \mid y, d)\,\mathrm{d}x. \tag{1}$$

Here, $\Omega_X$ denotes the sample space for the random variable $X$. For a classic introduction to the area of applied decision theory, see the book [2] by Raiffa and Schlaifer.

There is a broad spectrum of different approaches one may take when trying to solve these types of optimisation problems. In the case that both the preferences and beliefs of the decision maker may be adequately described by very simple functions $u$ and $\pi$, it might be possible to evaluate the integral in Eq.

[1] In this document, $\pi$ is used as general symbol for distributions. The idea behind this notation is that the arguments of $\pi$ indicates the set of random variables that we are concerned with in any particular instance.

(1) analytically. The problem would then be reduced to performing a (typically non-linear) optimisation. However, it is often the case that the integrand is too complicated to allow for an analytical integration. It is then necessary to turn to some form of numerical computation. Two widely employed methods are numerical quadrature and Monte Carlo integration. Quadrature methods have the advantage over Monte Carlo integration in that they often lead to more exact results, since there is no stochasticity involved in the computation. Moreover, deterministic error bounds are obtainable for certain methods. On the other hand, quadrature methods tend to be computationally very expensive for all but the smallest number of dimensions of the space $\Omega_X$. For a recent review of algorithms for Bayesian optimal design describing these and other approches, see [3] by Ryan et al.

This package uses Monte Carlo integration to evaluate the expected value in Eq. (1). There are two main reasons for taking this approach. Firstly, the aim has been to allow for some generality in the specification of the probabilistic model and utility model for the decision maker. In particular, it should be possible to specify models where $\Omega_X$ does not necessarily need to be of a small dimension. Secondly, the widely used **JAGS** software package (see [4]) already provides functionality for MCMC sampling from a wide class of models. The main strategy of this package is therefore to use **JAGS** to obtain a sample $(X_i)_{i=1}^n$ from the conditional distribution of $X$ given $Y = y$ and $d$ and estimate $f(d)$ with the sample mean

$$\hat{f}(d) = \frac{1}{n} \sum_{i=1}^{n} u(X_i, y, d).$$

# 2 Implementation overview

## 2.1 Single stage decision problems

There are three different ways in **bdpopt** to optimise the expected utility for a single stage decision problem using the MCMC simulation functionality provided by **JAGS**, each of which is described in the following subsections. The first approach is the most flexible, but requires that the user specifies a probabilistic model by means of a correctly written model file in the BUGS language. A partial **JAGS** data file must also be written, specifying the values of all model parameters that are not part of the decision set. The utility function may be any arbitrary R function (for which `formals` may be used to extract the argument names), but the argument names must constitute a subset of the names used in the BUGS model and data files. For the other two approaches, the probabilistic model has been fixed to specific BUGS files included in the package. The utility functions also have a fixed form, but allows for some flexibility since it is possible for the user to specify certain parameter values during the creation of the models.

### 2.1.1 JAGS model specified by user

The **JAGS** software may be called from R using the interface package **rjags**. The user specifies a probabilistic model in a model file, which is written in the BUGS language. In our setting, this model file defines a joint distribution for $(X, Y)$ given $d$. The user may condition on a specific value $y$ of $Y$ and decision

$d$ by specifying their fixed values in a so called data file (or, alternatively, using a list data structure in R). **JAGS** is then used to draw MCMC samples from the posterior distribution of $X$ given $Y = y$ and $d$. $Y = y$ is fixed during the optimisation, but since $d$ is varied over $D$ and the distribution of $X$ may depend on $d$ a new **JAGS** model is set up and samples drawn independently for each value of $d$. The Monte Carlo integration is done for each point in a grid $G$, which is defined as a subset of $D$. This will lead to the sample mean approximation $\hat{f}(d)$ of the true expected utility $f(d)$ for $d \in G$.

Now suppose that the samples $(X_i)_{i=1}^n$ drawn from $\pi(x \mid y, d)$ are independent and identically distributed random variables. Under appropriate conditions on $\pi(x \mid y, d)$ and $u(x, y, d)$, the Central Limit Theorem then implies that $(\hat{f}(d))_{d \in G}$ is a collection of independent and approximately normally distributed random variables,

$$\hat{f}(d) \sim \mathrm{N}\left(f(d), \frac{\mathrm{Var}(u(X, Y, d) \mid Y = y, d)}{n}\right), \tag{2}$$

with $n$ being the number of random draws in the simulation. However, the MCMC samples produced by **JAGS** are not independent (there exists autocorrelation in the chain). This implies that the formula in Eq. (2) is not directly applicable. General ergodic results from MCMC theory implies that $\hat{f}(d)$ converges (almost surely) to $f(d)$ as $n \to \infty$, but the autocorrelation of the chain means that the method used to estimate the variance of the sample mean must be modified. **bdpopt** uses the function `spectrum0.ar` in the **coda** package to obtain such an estimate.

The information that $(\hat{f}(d))_{d \in G}$ provides about the true form of $f(d)$ increases with the size of the grid and the sample size $n$ used for the sample means. Increasing the total number of grid points $g = |G|$ will decrease the risk of not including a grid point close to the true value of $d^*$. Increasing $n$ will lead to a better precision for the estimates, which decreases the risk that a suboptimal estimate of $d^*$ is found because of the stochastic variation of the simulations. The computation time increases linearly with $g$ and $n$, at least for moderately large values.

In many cases the expected utility surface is very flat close to the optimal $d^*$. This will make the stochastic variation of the estimates $\hat{f}(d)$ especially problematic. There are many suggestions in the literature on how to alleviate this problem. The approach chosen by the author follows the one described by Müller and Parmigiani in [5]. A smooth regression function $r(d)$ is fitted to the Monte Carlo samples $(\hat{f}(d))_{d \in G}$. This is done in **bdpopt** using either Gaussian process regression (GPR) (see, e.g., the book [6] by Rasmussen and Williams for an introduction to GPR) or local polynomial regression, via the function `loess` in the **stats** package.

GPR is a nonparametric method, which allows for flexibility when fitting the regression function. This is an important property, since a goal with the **bdpopt** package has been to allow the user to have the freedom to specify utility functions of an arbitrary form. However, the approach certainly has its limitations. Package testing by the author indicates that the GPR regression step works adequately only when the true function $f(d)$ is sufficiently smooth and there are not great differences in the rate of change for different regions of the domain $D$. Another thing for the user to keep in mind is that the computational

complexity of fitting the GPR model to the data, and also the complexity of the resulting regression function, increases with the number of grid points $g$.

By definition, a Gaussion process is a collection of random variables for which any finite subcollection have a joint Gaussian distribution. Such a process is completely specified by its mean and covariance functions. When the GPR is performed in **bdpopt**, the object that is modelled as a Gaussian process is the collection of sample means $(\hat{f}(d))_{d\in D}$. The mean and covariance functions for this process are defined as

$$m(d) = \mathbb{E}\left[\hat{f}(d)\right], \quad c(d,d') = \mathrm{Cov}\left(\hat{f}(d), \hat{f}(d')\right). \tag{3}$$

When the regression is performed by the function `fit.gpr`, it is done under the assumption that $m(d) = 0$[2]. Denoting the components of a vector decision $d$ of dimension $q$ by $d_1, \ldots, d_q$, the covariance function is assumed to be of the squared exponential type

$$c(d,d') = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{i=1}^{q}\left(\frac{d_i - d_i'}{l_i}\right)^2\right) + \mathbb{I}(d = d')\sigma_d^2. \tag{4}$$

The vector $(\sigma_f, l_1, \ldots, l_q)$ is referred to as the vector of hyperparameters for the GPR model. $\sigma_f$ sets the overall scale for the values of $\hat{f}(d)$. The parameters $l_1, \ldots, l_q$ play the roles of characteristic length scales for the different components of the vector decision. In general, large values for the length parameters leads to a large correlation between distant decisions and a smoother regression fit than for small values. $\sigma_d^2$ depends on the decision $d$ and is included in the covariance via the indicator factor only if $d' = d$. It may be interpreted as the variance of an error term that is assumed to be part of the observation $\hat{f}(d)$. The variance $\sigma_d^2$ is estimated using the function `spectrum0.ar` in the **coda** package. The regression performed by the function `fit.gpr` consists of maximising the marginal likelihood for the observed values $(\hat{f}(d))_{d\in G}$ with respect to the hyperparameters.

After the regression function $r(d)$ has been fitted to the data $(\hat{f}(d))_{d\in G}$, an approximation of $d^*$ is found by maximising $r(d)$. This is done by calling the `optim` in the **stats** package. Should the regression fail, the user also has the option to obtain an estimated $d^*$ by a direct comparison of the values of $\hat{f}(d)$ at the grid points. In some cases, it is a good strategy to perform a direct optimisation over the grid points using a broad and sparse grid as a first step in order to find a rough estimate of $d^*$. Provided that $f(d)$ is sufficiently smooth in the neighbourhood of the rough estimate, a finer grid may then be selected covering this neighbourhood. A regression and subsequent optimisation may then be done.

### 2.1.2 Simple normal model for phase III sample size optimisation

For this model the decision maker is taken to be a sponsor for a phase III clinical trial. The sponsor is the agent paying for the trial and we will assume it to be a

[2]If the regression is done over a grid where $m(d)$ is far from 0, then the user may perform an initial evaluation over the grid and select some appropriate non-zero constant $m_0$ as a better approximation. Regression may then be performed for the modified utility function obtained by subtracting $m_0$.

pharmaceutical company that performs the clinical trial in order to demonstrate the efficacy of the new drug for a regulatory authority. The regulatory authority examines the outcome of the trial and decides if there is enough evidence for market approval.

The function `n.opt` provides an interface to a simple model for the expected gain of a clinical trial sponsor faced with the situation of deciding on the optimal sample size for a confirmatory phase III trial. The sponsor pays a cost that is a linear function of the sample size, and obtains a fixed gain if a regulatory authority decides to approve the treatment for marketing. The decision of the authority is assumed to be based solely on the frequentist criterion of statistical significance.

The probabilistic model is defined as follows. There is a single response variable, $X$, which may be interpreted as an efficacy or clinical utility[3] response. $X$ is assumed to be normally distributed according to

$$X \mid \mu \sim \mathrm{N}\left(\mu, \sigma^2/n\right),$$

where $\mu$ is the true, unknown population mean for the response, $\sigma$ is the population standard deviation (assumed known) and $n$ is the sample size. Hence, $X$ may be interpreted as the sample mean of a sequence of i.i.d. random variables $X_1, \ldots, X_n$ such that $X_i \mid \mu \sim \mathrm{N}(\mu, \sigma^2)$, $i = 1, \ldots, n$. A conjugate normal prior with prior mean $\nu$ and prior standard deviation $\tau$ is assumed for the true population mean, $\mu \sim \mathrm{N}\left(\nu, \tau^2\right)$, leading to a prior predictive distribution for $X$ that is normal with mean $\nu$ and variance $\tau^2 + \sigma^2/n$.

The utility function for this model has the fixed form

$$u(X, \mu, n) = (G_a + F_a(X, \mu))\ \mathbb{I}\left(\frac{X}{\sqrt{\sigma^2/n}} > z_\alpha\right) - (C_f + C_s n). \tag{5}$$

The total gain for the sponsor in case of regulatory approval equals the sum of $G_a$ and $F_a(X, \mu)$. $G_a$ is a constant whereas the value of $F_a(X, \mu)$ depends on the trial outcome $X$ and the true value of the population mean $\mu$. $C_f$ is the fixed cost of setting up the trial and $C_s$ is the marginal cost per observation. The level for the one-sided approval test for statistical significance is $\alpha$ and $z_\alpha$ is defined by $z_\alpha = \Phi^{-1}(1-\alpha)$. The user may specify the values for the parameters of the model when calling `n.opt`.

### 2.1.3 Normal model with Emax dose responses for phase III dose and sample size optimisation

This model is specified in the BUGS file `normal_model_jags_model.R`, which is included with the package **bdpopt** in the external data folder `extdata`. The purpose of this fixed model is to provide an interface for phase III clinical trial optimisation with respect to dose and sample size given the results from a completed phase II trial. In the phase II trial, $k_2$ groups of patients have been given a new treatment, where the dose and sample size for group $i$ are denoted by $d_{2,i}$ and $n_{2,i}$, for $i = 1, \ldots, k_2$. It is assumed that the results of the phase II

[3]Clinical utility is a measure that somehow combines the efficacy and safety aspects of a treatment. For example, a clinical utility response might be constructed as an appropriately weighted linear combination of an efficacy and safety response.

trial may be summarised as one efficacy and one safety response for each patient included. For each group $i$, these responses are combined into a sample mean $Y_{2,i}^E$ for the effiacy responses and a sample mean $Y_{2,i}^S$ for the safety responses. The vectors of responses over all groups are denoted by $Y_2^E$ and $Y_2^S$.

The true population means $\mu_E$ and $\mu_S$ for the efficacy and safety responses depend on the dose $d$ and are assumed to follow Emax models,

$$\mu_E = \theta_1 + \theta_2 \frac{d^{\theta_4}}{\theta_3^{\theta_4} + d^{\theta_4}}, \tag{6}$$

$$\mu_S = \eta_1 + \eta_2 \frac{d^{\eta_4}}{\eta_3^{\eta_4} + d^{\eta_4}}, \tag{7}$$

where the unknown parameter vectors $\theta = (\theta_1, \theta_2, \theta_3, \theta_4)$ and $\eta = (\eta_1, \eta_2, \eta_3, \eta_4)$ determine the shapes of the Emax curves.[4] The priors for $\theta_1$, $\theta_2$, $\eta_1$ and $\eta_2$ are taken to be normal, whereas the priors for $\theta_3$, $\theta_4$, $\eta_3$ and $\eta_4$ are assumed to be log-normal.

By assumption, given $\theta$, $\eta$, the doses $d_2 = (d_{2,i})_{i=1}^{k_2}$ and the sample sizes $n_2 = (n_{2,i})_{i=1}^{k_2}$, the efficacy and safety responses are independent for each group and responses belonging to different groups are also independent. Moreover, both the efficacy and safety response for each patient are assumed to be normally distributed, with known sample variances given by $\sigma_E^2$ and $\sigma_S^2$. It follows that the conditional joint density over all phase II data may be split into factors according to

$$\pi\left(Y_2^E, Y_2^S \mid \mu_E\left(d_2, \theta\right), \mu_S\left(d_2, \eta\right)\right) = \prod_{i=1}^{k_2} \pi\left(Y_{2,i}^E \mid \mu_E\left(d_{2,i}, \theta\right)\right) \pi\left(Y_{2,i}^S \mid \mu_S\left(d_{2,i}, \eta\right)\right),$$

where

$$Y_{2,i}^E \mid \mu_E\left(d_{2,i}, \theta\right) \sim \mathrm{N}\left(\mu_E\left(d_{2,i}, \theta\right), \frac{\sigma_E^2}{n_{2,i}}\right), i = 1, \ldots, k_2,$$

$$Y_{2,i}^S \mid \mu_S\left(d_{2,i}, \eta\right) \sim \mathrm{N}\left(\mu_S\left(d_{2,i}, \eta\right), \frac{\sigma_S^2}{n_{2,i}}\right), i = 1, \ldots, k_2.$$

In phase III, the decision to be taken is on a dose $d_3$ and a sample size $n_3$. $k_3$ parallel and independent trials are then performed, each of which uses the same dose and sample size. The expected value of a particular choice is evaluated with respect to the posterior distribution of $\theta$ and $\eta$ given the phase II data (i.e., given $d_2$, $n_2$, $Y_2^E$ and $Y_2^S$). The independence structure and distributional assumptions for phase III are analogous to those of phase II, giving

$$\pi\left(Y_3^E, Y_3^S \mid \mu_E\left(d_3, \theta\right), \mu_S\left(d_3, \eta\right)\right) = \prod_{i=1}^{k_3} \pi\left(Y_{3,i}^E \mid \mu_E\left(d_3, \theta\right)\right) \pi\left(Y_{3,i}^S \mid \mu_S\left(d_3, \eta\right)\right),$$

[4]The typical names corresponding to the first three component of $\theta$ and $\eta$ are: $E_0$ for $\theta_1$ and $\eta_1$, $E_{\max}$ for $\theta_2$ and $\eta_2$, $ED_{50}$ for $\theta_3$ and $\eta_3$.

with

$$Y_{3,i}^E \mid \mu_E(d_3,\theta) \sim \mathrm{N}\left(\mu_E(d_3,\theta), \frac{\sigma_E^2}{n_3}\right), i = 1,\ldots,k_3,$$
$$Y_{3,i}^S \mid \mu_S(d_3,\eta) \sim \mathrm{N}\left(\mu_S(d_3,\eta), \frac{\sigma_S^2}{n_3}\right), i = 1,\ldots,k_3.$$

The utility function for this model has the form

$$u = G_a R - (C_f + C_s n_3 k_3), \tag{8}$$

where $G_a$ is the gain upon regulatory approval, $R$ is an indicator function for regulatory approval, $C_f$ is a fixed cost of setting up the trials and $C_s$ is the cost per observation. $G_a$ is defined as

$$G_a(Y_3^E, Y_3^S, \mu_E, \mu_S, g_E, g_S, p) =$$
$$p\left(\frac{g_E}{k_3}\sum_{i=1}^{k_3} Y_{3,i}^E + \frac{g_S}{k_3}\sum_{i=1}^{k_3} Y_{3,i}^S\right) + (1-p)(g_E\mu_E + g_S\mu_S). \tag{9}$$

$g_E$ is a constant factor giving the utility per efficacy unit, and $g_S$ is a constant factor giving the utility loss per safety unit (so if safety units are positive, $g_S$ should be negative). $p \in [0,1]$ is a constant that weighs the relative importance of the responses observed in the trial and the true population means. The indicator function for approval, $R$, is defined to be 1 if and only if the sample size in each trial is at least $n_{\min}$, a one-sided statistical significance can be shown independently for efficacy in each trial at the level $\alpha$ and a one-sided statistical significance can be shown independently for $Y_{S,i}^{III} - m_S$ in each trial at the level $\alpha$. $m_S$ may be interpreted as a maximum safety level. This means that $R$ may be written as

$$R(Y_3^E, Y_3^S, n_3, n_{\min}, \alpha, m_S) =$$
$$\mathbb{I}(n_3 > n_{\min}) \prod_{i=1}^{k_3} \mathbb{I}\left(Y_{3,i}^E > \frac{z_\alpha \sigma_E}{\sqrt{n_3}}\right) \prod_{i=1}^{k_3} \mathbb{I}\left(Y_{3,i}^S < m_S - \frac{z_\alpha \sigma_S}{\sqrt{n_3}}\right). \tag{10}$$

## 2.2 Sequential decision problems

**bdpopt** supports two ways to solve sequential decision problems. For the first and more flexible alternative, the user must specify all the components defining the sequential problem. The second alternative provides an interface to a very specific group sequential model with normal responses for which the probabilistic model, the decisions available at each stage and the form of the utility functions have already been fixed.

### 2.2.1 Full model specification by the user

In addition to the one-stage optimisation procedure described in the previous section, **bdpopt** also provides some basic functionality for solving certain types of sequential decision problems, the form of which will now be described.

Let $k$ denote the number of stages in a given decision problem. For $i = 1,\ldots,k$, there is a nonempty set of decisions $D_i$ available for selection. It

is assumed that each set $D_i$ may be partitioned into three disjoint sets, $D_i^c$, $D_i^t$ and $D_i^o$, which will be referred to as the continuation decisions, terminal decisions and terminal observation decisions for the stage. In each stage $i$, the decision maker selects an element $d_i$ belonging to one of these three types of decision sets. If $d_i \in D_i^c$, then the value of a random variable $X_i$ is observed, a utility value $u_i^c(d_i, X_i)$ is collected and the process proceeds to the next stage. If $d_i \in D_i^t$, no observation is made, a utility value $u_i^t(d_i, \theta)$ is collected and the decision process is terminated. If $d_i \in D_i^o$, an observation $X_i$ is made, a utility value $u_i^o(d_i, X_i, \theta)$ is collected and the decision process is terminated.

In addition to the random variables $(X_i)_{i=1}^k$ associated with each stage, there is also an unknown parameter $\theta$ associated with the decision problem. The information available to the decision maker concerning $\theta$ before the first stage is summarised in terms of a prior distribution $\pi(\theta)$. It is assumed that, given $\theta$, the realisations of the random variables $X_1, \ldots, X_k$ are independent. More precisely, it is assumed that the joint distribution of $(X_1, \ldots, X_k)$ given $\theta$ and $d_1, \ldots, d_k$ may be split into factors according to

$$\pi\left(X_1, \ldots, X_k \mid \theta, d_1, \ldots, d_k\right) = \prod_{i=1}^{k} \pi_i\left(X_i \mid \theta, d_i\right). \tag{11}$$

Note the stage subscripts attached to the distributions on the right hand side of Eq. (11), which highlights the possibility that the distributions for the observations may depend on the stage in addition to the decision taken at that stage.

The computational strategy used to solve the sequential decision problem combines backward induction with simulation. In order to make this approach at all viable for solving problems involving more than just a few stages, the exponential growth in the computations required to uphold stagewise optimality when performing the backward induction must be dealt with. To illustrate the problem, suppose that the decision maker, being at stage $i$, has made the decisions $d_1, \ldots, d_{i-1}$ and observed the stage-wise outcomes $X_1 = x_1, \ldots, X_{i-1} = x_{i-1}$. The task is now to choose $d_i$ optimally, given that, whatever particular outcome $X_i$ is observed, the future decisions beyond stage $i$ are chosen optimally. Letting $v_i\left((x_j)_{j=1}^i, (d_j)_{j=1}^i\right)$ be an estimate of the expected utility of continuing optimally from stage $i+1$ and onwards, the problem to solve is

$$\begin{aligned}
&\arg\max_{d_i \in D_i} \int_{\Omega_{X_i}} v_i\left((x_j)_{j=1}^i, (d_j)_{j=1}^i\right) \pi_i(x_i \mid (x_j)_{j=1}^{i-1}, (d_j)_{j=1}^i)\, \mathrm{d}x_i \iff \\
&\arg\max_{d_i \in D_i} \iint_{\Omega_{X_i} \times \Omega_\Theta} v_i\left((x_j)_{j=1}^i, (d_j)_{j=1}^i\right) \pi_i(x_i \mid \theta, d_i) \pi(\theta \mid (x_j)_{j=1}^{i-1}, (d_j)_{j=1}^i)\, \mathrm{d}x_i\, \mathrm{d}\theta.
\end{aligned}$$

The value of the integral in the equation above can be estimated by simulating first from the posterior distribution of $\theta$ given the previous observations and decisions and then from the conditional distribution of $X_i$ given $\theta$ and $d_i$. The computational problem stems from the fact that $v_i\left((x_j)_{j=1}^i, (d_j)_{j=1}^i\right)$ must be available for all possible histories of observations and decisions. If, say, a common grid $G_X$ is used to save the possible values for the observations, and $D$ is the same for all stages, then $|G_X|^i |D|^i$ values of $v_i$ must be available at stage $i$.

In order to deal with the exponential increase in the computations and memory required to perform a straightforward backward induction, the **bdpopt** package uses the method described by Brockwell and Kadane in [7]. The major additional assumption made to make the computations feasible is that the information regarding the parameter $\theta$ provided by the previous observations $x_1, \ldots, x_i$ and the decisions $d_1, \ldots, d_i$ may be completely summarised by means of a state vector $s_i$. It is assumed that $s_i$ belongs to a space of finite dimension, $S$, and that this space is the same for all stages. This implies that the posterior distribution of $\theta$ at each stage may be parameterised in terms of some value in $S$. Therefore, the expected utility $v_i$ of continuing optimally becomes a function of $s_i$ only. The backward induction step at stage $i$ may then be written as

$$\arg\max_{d_i \in D_i} \iint_{\Omega_{X_i} \times \Omega_\Theta} v_i\left(s_i = t_i(d_i, s_{i-1}, x_i)\right) \pi_i(x_i \mid \theta, d_i) \pi(\theta \mid s_{i-1}) \, \mathrm{d}x_i \, \mathrm{d}\theta, \qquad (12)$$

where $t_i$ denotes a transfer function, taking a decision, a state $s_{i-1}$ at stage $i$ and an observation $x_i$ into a new state $s_i$ at stage $i+1$. This transfer function may be viewed as a component of the probabilistic model and must be specified in an appropriate way by the user. To estimate the integral in Eq. (12), simulation is done first from the distribution of $\theta$ given $s_{i-1}$ and then from the distribution of $X_i$ given $\theta$ and $d_i$. Hence, in order to fully specify the probabilistic model the user must specify the form of the transfer functions $(t_i)_{i=1}^{k-1}$ and provide R functions implementing the simulation from $\pi(\theta \mid s)$, $s \in S$, and from $\pi_i(x_i \mid \theta, d_i)$, $i = 1, \ldots, k$, $d_i \in D_i^c \cup D_i^o$.

The backward induction is implemented by computing the optimal expected utility and corresponding action on a grid $G_S$ in $S$ for each stage. This implies that the total computational effort required is proportional to the product $k|G_S|$. The time required to perform the computation also grows linearly with the total number of simulation iterations used when estimating the expected utilities. The quality of the approximation increases as the volume of the boundary of $G_S$ increases, as the distance between the grid points decreases and as the number of simulation samples increases.

### 2.2.2 Group sequential normal model

This model again takes the viewpoint of a clinical trial sponsor aiming for regulatory approval. The sponsor collects evidence about a treatment in $k$ stages. At each stage, there are precisely two decisions available. For all but the last stage, the options are to either continue and take a new sample of group size $n$ or to stop (abort) the process. To proceed from a stage $i$ the sponsor has to pay a stage cost covering the expenses required to collect the responses from $n$ patients and combine them into a stage response $X_i$ for the group. If the sponsor instead decides to stop, no cost is incurred (but any potential future gain is lost). The evidence collected in these preliminary stages is not presented to a regulatory authority deciding on approval, but is only used by the sponsor to increase its knowledge about the true value $\theta$ of the population mean of the efficacy response for the treatment.

In the last stage, the sponsor may decide to either abort the decision process or to file an application for approval. It it decides to abort, the net utility of the last stage is 0. If it decides to file the application, the net utility is taken to

be the difference between a gain proportional to $\theta$ and a final investment cost. Hence, the last stage is the only part of the decision process in which the sponsor may collect a positive contribution to the total utility. This contribution is taken to be proportional to $\theta$ because it is reasonable to expect that the probability of approval and the potential sales after approval are both increasing functions of $\theta$.

At each stage, the conditional distribution of the observation $X_i$ is given by $X_i \mid \theta \sim \mathrm{N}(\theta, \sigma^2/n)$, where $\sigma$ is a known population standard deviation and $n$ is the sample size per group. Hence, $X_i$ may be interpreted as the sample mean of a sequence of $n$ i.i.d. normal random variables given $\theta$. The prior for $\theta$ before the first stage is taken to be a conjugate normal distribution with known prior standard deviation $\tau$. With this setup, by standard conjugate updating, it may easily be shown that the posterior distribution of $\theta$ at stage $i$ is normal, with variance given by

$$\left(\frac{1}{\tau^2} + \frac{i-1}{\sigma^2/n}\right)^{-1}. \tag{13}$$

Since all of the quantities in the expression for the posterior variance are known, it follows that the posterior distribution for $\theta$ at a given stage is completely characterised by the posterior mean. The state variable $s$ for this model is therefore taken to be the posterior mean for $\theta$, giving a state space $S = \mathbb{R}$.

# 3 General workflow

This section contains step by step instructions on how to set up the different model types and perform optimisation using the interfaces provided by the package.

## 3.1 Single stage decision problems

### 3.1.1 JAGS model specified by user

1. Write a model file (henceforth referred to as `model.R`) and a partial data file (henceforth referred to as `data.R`) specifying the probabilistic model. The decision variables of the problem are implicitly defined as the additional names that would have to be defined in order to make `model.R` plus `data.R` a complete **JAGS** model.

2. Create a simulation model object by calling the function `sim.model` with arguments `model.R` and `data.R`. Alternatively, the contents of the data file may also be supplied as a named list of R objects.

3. Define a utility function `u`. The argument names must constitute a subset of the names used in the BUGS model and data files.

4. Create a grid specification list object defining the names of the decision variables and the extent and step size of the grid. Such an object should consists of a named list of grid specifications for the individual variables. Each grid specification should be a list of two components. The first component is a dimension vector, which specifies the dimension of the array value assumed by the decision variable at a grid point. The second

component should be a list of vectors of length equal to the total number of elements of an array value (i.e., equal to the product of the elements of the dimension vector). Each such vector must have the form `c(lower, upper, step)`. These vectors are passed to `seq` in order to generate a range of values for each component of the array.

5. Evaluate the model on the grid by calling the function `eval.on.grid`, passing the previously constructed simulation model, utility function `u` and grid as arguments. This is the step in which MCMC samples are produced by calling **JAGS**. The results of the simulation are saved in a new model object returned by `eval.on.grid`.

6. Fit a regression function to the grid points by calling the function `fit.gpr` or `fit.loess` with the model object obtained in the previous step as an argument. A new model object containing the regression function will be returned. This step is optional, since optimisation may also be performed directly over the grid without trying to fit a smooth regression function first. However, it is required if any option other than "`Grid`" is to be passed to the optimisation function `optimise.eu`.

7. Optionally, inspect the results from the simulation and regression steps by calling the generic function `plot` with the model object returned from `eval.on.grid`, `fit.gpr` or `fit.loess` as an argument.

8. Optimise directly over the grid or using the regression function by calling `optimise.eu`. The (approximately) optimal decision and corresponding optimal utility will be returned as two components in a list.

#### 3.1.2 Simple normal model for phase III sample size optimisation

The interaction with this model consists of a single step:

1. Call `n.opt` to perform evaluation on a one-dimensional grid for the sample sizes, followed by an optimisation over the grid and an optional plotting of the results. The grid points and corresponding simulated expected utility values are returned together with the optimal sample size and corresponding expected utility as components in a list.

#### 3.1.3 Normal model with Emax dose responses for phase III dose and sample size optimisation

1. Create a normal model object. This may be done in two different ways. The first alternative is to call the function `create.normal.model`. The parameters defining the probabilistic model must then be passed as arguments. The second alternative is to call `create.normal.model.from.file`, which loads the model parameters from the file `normal_model_jags_data.R`. Alternative models may then be specified by changing the contents of this file.

2. Create a utility function by calling `create.utility.function`. This function takes the model object created in the previous step as an argument. The user must also pass values defining the parameters of the

utility function as arguments to the function, $g_E$ (`cE`), $g_S$ (`cS`), $p$ (`p`), $m_S$ (`safety.max`), $C_f$ (`fixed.cost`) and $C_s$ (`cost.per.sample`).

3. At this point a model object and a utility function have been specified. The application of the functions `eval.on.grid`, `fit.gpr`, `fit.loess` and `optimise.eu` for evaluation on a grid, fitting of a regression function and optimisation now proceeds just as for the case of a general **JAGS** model fully specified by the user.

## 3.2 Sequential decision problems

### 3.2.1 Full model specification by the user

1. Choose a value for `n.stages`, the number of stages for the decision problem.

2. Construct a specification of the probabilistic model in terms of functions `post.sample`, `pred.sample` and `update.state`. `post.sample` should provide independent samples from the posterior distribution of the parameter $\theta$ given the current stage and state $s$. `pred.sample` should provide independent samples from the predictive distribution of a new observation $X$ at a given stage, given $\theta$ and the decision taken. `update.state` should take the current stage, the state $s$, a decision $d$ and a list of observed values into a corresponding list of updated state values for the next stage.

3. Construct a specification of the decisions available at each stage in terms of lists `cont.decisions`, `term.decisions` and `term.obs.decisions`, corresponding to $(D_i^c)_{i=1}^k$, $(D_i^t)_{i=1}^k$ and $(D_i^o)_{i=1}^k$, respectively. The length of each decision list must be equal to the number of stages $k$. The $i$:th element of each list should be a list of the decisions available (of the respective type) at stage $i$. Note that, for each stage $i$, at least one of the sets $D_i^c$, $D_i^t$ and $D_i^o$ must be nonempty. Also, for the last stage $k$, there can be no continuation decisions (i.e., $D_k^c$ must be empty).

4. Construct a specification of the utility model at each stage in terms of lists `cont.decisions`, `term.decisions` and `term.obs.decisions`, corresponding to $((u_i^c(d_i, X_i))_{i=1}^k$, $(u_i^t(d_i, \theta))_{i=1}^k$ and $(u_i^o(d_i, X_i, \theta))_{i=1}^k$, respectively. If at stage $i$ a certain type of decision is not available, that is, if the $i$:th list in `cont.decisions`, `term.decisions` or `term.obs.decisions` is empty, then the corresponding element in `cont.decisions`, `term.decisions` or `term.obs.decisions` may be set to `NA`.

5. Create a sequential decision problem object by calling the function `sequential.dp` with the objects constructed in the previous steps as arguments.

6. Define a grid for the state $s$ as a subset of $S$ in terms of three numeric, atomic vectors `mins`, `maxs` and `steps`. Each must be of length equal to the dimension of $S$. `mins` contains the minimum values of the grid points in each dimension, `maxs` the maximum values and `steps` the step sizes between grid points in each dimension.

7. Solve the sequential problem by calling `optimise.sequential.eu` with the objects constructed in the previous steps passed as arguments. `optimise.sequential.eu` has an optional argument with name `state.start`. If left unspecified, it is set to `NA`, and the output from `optimise.sequential.eu` will consist of a list with two components. The first is a function taking a stage and state into the optimal decision for the closest grid point in $S$ and the second is a function taking the stage and a value $s$ into the optimal utility for the closest grid point. In case a value for `state.start` is provided, then the optimal action and utility will be computed only for the specified value for the first stage, and the output will consist of a list of four components. The first two components gives the optimal decision and expected utility for stage 1, and the remaining two components are as for the case `state.start = NA`.

#### 3.2.2 Group sequential normal model

1. Create a sequential normal decision problem object by calling the function `sequential.normal.dp`. The user must specify the number of stages (`n.stages`), the group size $n$ (`group.size`), the standard deviation parameters $\tau$ and $\sigma$ (`tau` and `sigma`) and the gain and cost parameters (`stage.cost`, `final.cost`, `final.gain`).

2. Solve the sequential normal decision problem by calling the function `optimise.sequential.normal.eu`, passing the decision problem object created in the previous step as an argument. The user must specify the range and step size of the grid for the state.

# 4 Examples

This section contains code examples illustrating how the different model types supported by the package may be set up and optimised. Note that `library(bdpopt)` must be called before any of the examples are run.

## 4.1 Simple normal model for phase III sample size optimisation

```
> ## Perform an optimisation for the simple normal model
> out <- n.opt(nu = 0, tau = 1, sigma = 1, alpha = 0.025,
+              gain.constant = 1, gain.function = function(X, mu) 0,
+              fixed.cost = 0, sample.cost = 0.005,
+              k = 1, n.min = 1, n.max = 50, n.step = 1,
+              n.iter = 10000, n.burn.in = 1000, n.adapt = 1000,
+              regression.type = "loess",
+              plot.results = TRUE, independent.SE = FALSE,
+              parallel = FALSE, path.to.package = NA)
> ## Print the grid points used for the sample size,
> ## and the corresponding estimates of the expected utility
> print(out$ns)
> print(out$eus)
> ## Print the estimate of the optimal sample size,
```

```
> ## and the corresponding utility
> print(out$opt.arg)
> print(out$opt.eu)
>
```

### 4.2 Normal model with Emax dose responses for phase III dose and sample size optimisation

```
> ## Mean and precision parameters for the priors
> theta.mu <- c(0, 2, 0, 0); theta.tau <- c(1, 1, 8, 8)
> eta.mu <- c(0, 2, 0, 0); eta.tau <- c(1, 1, 8, 8)
> ## Sample size and doses for each observation in phase II
> n.II <- rep(10, 10); d.II <- seq(0.1, 1, 0.1)
> ## Observed responses phase II responses,
> ## taken from the efficacy and safety models using the parameter values
> ## theta = eta = c(0, 2, 1, 1) (rounded to two decimals).
> YE.II <- c(0.18, 0.33, 0.46, 0.57, 0.67, 0.75, 0.82, 0.89, 0.95, 1.00)
> YS.II <- c(0.18, 0.33, 0.46, 0.57, 0.67, 0.75, 0.82, 0.89, 0.95, 1.00)
> sigmaE <- 1; sigmaS <- 1 ## Standard deviations
> k.III <- 2 ## Number of phase III trials
> m1 <- create.normal.model(theta.mu, theta.tau, eta.mu, eta.tau,
+                           n.II, d.II, YE.II, YS.II,
+                           sigmaE, sigmaS, k.III, path.to.package = NA)
> ## Define a utility function
> n.min <- 0; sig.level<- 0.025; safety.max <- 0.6
> cE <- 1300; cS <- -1000; p <- 0.5
> fixed.cost <- 10; cost.per.sample <- 0.2
> u <- create.utility.function(m1, n.min, sig.level, safety.max,
+                              cE, cS, p, fixed.cost, cost.per.sample)
> ## Define a grid and simulate the utility for each grid point
> n.iter <- 4000; n.burn.in <- 1000; n.adapt <- 1000
> gsl <- list(n.III = list(c(1), list(c(10, 150, 10))),
+             d.III = list(c(1), list(c(0.1, 0.4, 0.1))))
> m2 <- eval.on.grid(m1, u, gsl, n.iter, n.burn.in, n.adapt,
+                    independent.SE = FALSE, parallel = TRUE)
> ## Do gaussian process regression for the model
> m3 <- fit.gpr(m2, start = c(30, 50, 0.2), gr = TRUE, method = "L-BFGS-B",
+                   lower = c(10, 10, 0.1), upper = Inf)
> ## Plot the results of the evaluation and gpr regression
> plot(m3, "n.III[1]", fixed = seq(0.1, 0.4, 0.1))
> ## Optimisation (defaulting to method "L-BFGS-B" of the optim function)
> optimise.eu(m3, start = c(100, 0.3))
>
```

### 4.3 Group sequential normal model

```
> ## Create a sequential decision problem object
> dp <- sequential.normal.dp(n.stages = 4, group.size = 10,
+                            tau = 1, sigma = 1,
+                            stage.cost = 0.1, final.cost = 1, final.gain = 2)
```

```
> ## Solve the sequential decision problem and plot the results
> out <- optimise.sequential.normal.eu(dp = dp,
+                                      range = 8, step.size = 0.02,
+                                      prior.mean = 0,
+                                      n.sims = 1000,
+                                      plot.results = TRUE)
> ## Print the optimal decision and corresponding expected utility
> ## at the first stage assuming a prior mean of 0
> print(out$opt.decision(1, 0))
> print(out$opt.utility(1, 0))
>
```

## Acknowledgements

This project has received funding from the European Union's 7th Framework Programme for research, technological development and demonstration under the IDEAL Grant Agreement no 602552.

## References

[1] IDEAL project, `http://www.ideal.rwth-aachen.de/`, accessed 2015-10-30.

[2] H. Raiffa, R. Schlaifer. *Applied statistical decision theory.* The M.I.T Press: Cambridge, MA, 1968.

[3] E. G. Ryan, C. C. Drovandi, J. M. McGree, A. N. Pettitt. *A Review of Modern Computational Algorithms for Bayesian Optimal Design.* International Statistical Review, 2015.

[4] M. Plummer, `http://mcmc-jags.sourceforge.net/`, accessed 2015-10-26.

[5] P. Müller, G. Parmigiani. *Optimal Design via Curve Fitting of Monte Carlo Experiments.* Journal of the American Statistical Association, Vol. 90, No. 432, 1995.

[6] C. E. Rasmussen, C. K. I. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, 2006.

[7] A. E. Brockwell, J. B. Kadane. *A Gridding Method for Bayesian Sequential Decision Problems.* Journal of Computational and Graphical Statistics, Volume 12, Number 3, Pages 566-584, 2003.