ICAOD: An R Package for Finding Optimal Designs for Nonlinear Models Using Imperialist Competitive Algorithm

Ehsan Masoudi
University of Münster

Heinz Holling
University of Münster

Weng Kee Wong
University of California, Los Angeles

Abstract

Optimal design ideas are increasingly used in different disciplines to rein in experimental costs. Given a statistical model and a design criterion, optimal designs determine the number of experimental points to observe the responses, the design points and the number of replications at each design point. Currently, there are very few free and effective computing tools for finding different types of optimal designs for a general nonlinear model, especially when the criterion is not differentiable. We introduce an R package **ICAOD** to find various types of optimal designs and they include locally, minimax and Bayesian optimal designs for different nonlinear models. Our main computational tool is a novel metaheuristic algorithm called imperialist competitive algorithm (ICA) and inspired by socio-political behavior of humans and colonialism. We demonstrate its capability and effectiveness using several applications. The package also includes several theory-based tools to assess optimality of a generated design when the criterion is a convex function of the design.

Keywords: $c-$optimality, $D-$optimality, design of experiments, evolutionary algorithm, optimization, population-based algorithm, R.

1. Introduction

Optimal designs have been extensively applied in many research studies to reduce the cost of experimentation. For instance, Holling and Schwabe (2013) provided examples in psychology and Dette, Kiss, Bevanda, and Bretz (2010) gave examples in dose-response studies. Further applications of optimal designs in engineering and epidemiology are described in Berger and Wong (2009), which also contains applications of optimal design ideas in other disciplines. Given a statistical model and an optimality criterion, optimal designs determine the optimal number of design points required, their locations to observe the responses and the number of replications required at each location. The optimality criterion should accurately reflect the objective of the study to the extent possible and is usually formulated as a scalar function of the Fisher information matrix (FIM) that measures the worth of the design (Lehmann and Casella 1998). For example, if the objective of a study is to estimate the model parameters as accurately as possible, D-optimality is often used. Such an optimal design maximizes the determinant of the FIM and is called D-optimal. When errors are independent and

normally distributed, D-optimal designs minimize the volume of the confidence ellipsoid of the model parameters by minimizing the generalized variance, i.e., the determinant of the variance-covariance matrix (Abdelbasit and Plackett 1983).

For nonlinear models, the FIM depends on the unknown model parameters to be estimated and so the design criterion cannot be directly optimized. There are different approaches to deal with this parameter dependency: a) *locally optimal designs:* These are found by replacing the unknown parameters with some initial estimates from a pilot or previous study (Chernoff 1953). Locally optimal designs usually become inefficient when the initial estimates are far from their true unknown values. b) *minimax optimal designs:* They minimize the maximum inefficiency over a user-selected parameter space (Sitter 1992). The optimal designs are conservative in that they protect the experiment from the worst case scenario that may happen from a poor choice of parameter values over a user-specified space of plausible values for the unknown parameters. Finding minimax optimal designs is complicated because it involves solving multi-level nested optimization problems and the objective function (minimax criterion) is not differentiable. c) *Bayesian optimal designs:* These optimal designs are found by optimizing an optimality criterion averaged over a user-specified (continuous) prior distribution for the unknown parameters (Chaloner and Larntz 1989; Chaloner and Verdinelli 1995; Atkinson 1996). Strictly speaking, the latter are not fully Bayesian because they do not involve computing a posterior distribution. Instead, they borrow the concept of having prior distributions to find robust designs for the frequentists (Graßhoff, Holling, and Schwabe 2012; Bürkner, Schwabe, and Holling 2019). Accordingly, they are sometimes referred to as "pseudo" Bayesian designs (Firth and Hinde 1997). In the optimal design literature, Bayesian optimal designs found under a discrete prior distribution are usually referred to as *robust* or *optimum-on-average* designs (Fedorov and Hackl 2012). For an overview of optimal designs for nonlinear models, see Fedorov and Leonov (2013).

There are several software packages to create and analyze design of experiment (DoE) for different purposes. For a review on statistical R packages in design of experiments, see `https://cran.r-project.org/web/views/ExperimentalDesign.html`. Only a few of them are able to find different types of optimal designs to deal with the parameter dependency for various nonlinear models. To the best of our knowledge, none of the available software packages, commercial or otherwise, provides an option to find minimax optimal designs for nonlinear models. For example, the R (R Core Team 2019) package **LDOD** (Masoudi, Sarmad, and Talebi 2013) finds locally $D-$optimal *approximate* designs for a large class of nonlinear models and the **acebayes** R package (Overstall, Woods, and Adamou 2017) determines a more general class of fully Bayesian *exact* designs using the approximate coordinate exchange algorithm (Overstall and Woods 2017). Likewise, the recently available **VNM** R package finds multiple-objective locally optimal designs for a specific model, i.e., the four-parameter Hill model commonly used in dose-response studies (Hyun, Wong, and Yang 2018). Among the commercial software, JMP® (SAS Institute Inc. 2016) can also find Bayesian $D-$optimal exact designs for nonlinear models.

This paper introduces the R package **ICAOD** (Masoudi, Holling, and Wong 2016) for finding a variety of optimal designs for nonlinear models using a novel metaheuristic algorithm called *imperialist competitive algorithm(ICA)*. This algorithm is inspired by socio-political behavior of humans (Atashpaz-Gargari and Lucas 2007; Hosseini and Al Khaled 2014) and is modified by Masoudi, Holling, and Wong (2017) to find optimal designs for nonlinear models. We believe that this **ICAOD** package is the first single self-contained statistical package that

presents a framework to find locally, minimax and Bayesian optimal designs for nonlinear models. Similar to many popular nature-inspired metaheuristic algorithms, such as particle swarm optimization (PSO) algorithm (Kennedy and Eberhart 1995), ICA does not have a rigorous proof of convergence (Yang 2011). When the criterion is a convex function on the set of design measures, equivalence theorems are available and the **ICAOD** package includes tools to confirm optimality of a design. More generally, the proximity of any design to the optimum without knowing the latter can be evaluated in terms of an efficiency lower bound. In particular, if this bound is unity, this confirms optimality of the design. This feature is useful to recognize a case of pre-mature convergence in optimal design problems.

Section 2 reviews the statistical setup and theory for finding optimal designs for nonlinear models. Section 3 describes the imperialist competitive algorithm (ICA) and Section 4 provides implementation details for the **ICAOD** package. In Section 5, we provide two examples to show the functionality of the **ICAOD** package; Section 5.1 finds locally and minimax $D-$optimal designs for a logistic model with application in educational testing and Section 5.2 presents optimum-on-average and Bayesian $D-$optimal designs for a sigmoid Emax model for dose-response studies. The **ICAOD** package was first written to find locally $D-$optimal designs, which are arguably most overused in practice, but it now also finds user-defined optimal designs. Section 6 illustrates how to use this feature to find $c-$optimal designs for a two-parameter logistic model in dose response studies. Section 7 concludes with a summary.

2. Background and Optimal Designs

Let $E(Y) = f(\boldsymbol{x}, \boldsymbol{\theta})$ be the mean of the response Y at the values of the independent variables $\boldsymbol{x}$ defined on a user-selected *design space* χ, and let f be a known function, apart from the model parameters $\boldsymbol{\theta} = (\theta_1, ..., \theta_p)^T$. Throughout we assume that there are resources to take N observations for the study and given an optimality criterion, we want to find the best choices for the levels of the independent variables to observe the outcome Y. There are two types of designs: exact and approximate. An *exact* design ξ_N on χ is defined by a set of k distinct levels $\boldsymbol{x}_i$,

$$\xi_N = \left\{ \begin{array}{cccc} \boldsymbol{x}_1 & \boldsymbol{x}_2 & ... & \boldsymbol{x}_k \\ n_1/N & n_2/N & ... & n_k/N \end{array} \right\}, \tag{1}$$

where $\boldsymbol{x}_j \in \chi$, n_j is the number of replications of $\boldsymbol{x}_j$ in the observations sample and $N = \sum_{j=1}^{k} n_j$. Here, $x_j, j = 1..., k$ are referred to as *support points* or *design points* of ξ_N. Given N and a specific design criterion, an optimal exact design finds the best value of k and the best values of $x_1, \ldots, x_k, n_1, \ldots, n_k$. Such optimization problems are notoriously difficult and in practice, we find optimal approximate designs instead. They are probability measure on χ are found independent of the sample size N. An approximate design ξ with k support points has the form

$$\xi = \left\{ \begin{array}{cccc} \boldsymbol{x}_1 & \boldsymbol{x}_2 & ... & \boldsymbol{x}_k \\ w_1 & w_2 & ... & w_k \end{array} \right\}, \tag{2}$$

where $w_j > 0$ is the proportion of observations that are assigned to $\boldsymbol{x}_j$ and $\sum_{j=1}^{k} w_j = 1$. They are implemented by first rounding each value of Nw_i to the nearest integer Nw_i^* subject to $Nw_1^* + \ldots + Nw_k^* = N$ and taking Nw_i* observations at $\boldsymbol{x}_i, i = 1, \ldots, k$. Some optimal rounding procedures are available in (Pukelsheim and Rieder 1992). When the criterion is convex, there are algorithms for finding many types of optimal approximate designs and

theory to confirm optimality of an approximate design. When the design is not optimal, a theory-based efficiency lower bound of the design is available to determine its proximity to the optimum, without knowing the optimum. For these reasons, we focus on optimal approximate designs found under a convex functional in the rest of the paper.

A design that minimizes a convex design criterion ψ over the space of all designs on ξ. This means that given the model and ψ, the optimal number of support points, k, the optimal support points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ and their corresponding $w_1, \ldots, w_k$ have to be determined. For example, if estimating model parameters is of interest, $D-$optimality, defined by the logarithm of the determinant of the inverse of the FIM, is a convex functional over the space of all designs on χ (Fedorov and Leonov 2013; Silvey 1980) and the resulting design is called $D-$optimal. In what follows, we focus on the $D-$optimality criterion. The description for other optimality criteria is very similar.

Assuming all observation errors are independent and normally distributed with means 0 and a constant variance $\mathrm{Var}(Y)$, the FIM of a generic k-point approximate design ξ is given by

$$M(\xi, \boldsymbol{\theta}) = \sum_{i=1}^{k} w_i I(\boldsymbol{x}_i, \boldsymbol{\theta}), \tag{3}$$

where

$$I(\boldsymbol{x}_i, \boldsymbol{\theta}) = \frac{1}{\mathrm{Var}(Y_i)} \nabla f(\boldsymbol{x}_i, \boldsymbol{\theta}) \nabla f(\boldsymbol{x}_i, \boldsymbol{\theta})^T,$$

and $\nabla f(\boldsymbol{x}_i, \boldsymbol{\theta})^T = \left(\frac{\partial f(\boldsymbol{x}_i, \boldsymbol{\theta})}{\partial \theta_1}, \cdots, \frac{\partial f(\boldsymbol{x}_i, \boldsymbol{\theta})}{\partial \theta_p} \right)$. Here, $\frac{\partial f(\boldsymbol{x}_i, \boldsymbol{\theta})}{\partial \theta_j}$ denotes the partial derivative of f with respect to θ_j. The FIM is singular if $k < p$. To avoid singular designs, i.e., designs with singular Fisher information matrices, we assume $k \geq p$.

Clearly, the FIM (3) depends on the unknown parameters for nonlinear models. Different approaches have been proposed to deal with this parameter dependency based on the type of information available for the unknown parameters. For example, let $\boldsymbol{\theta}_0$ be an initial guess for $\boldsymbol{\theta}$ available from a similar study. A locally $D-$optimal design ξ^*_{loc} minimizes

$$\psi_{loc}(\xi) = -\log |M(\xi, \boldsymbol{\theta}_0)|, \tag{4}$$

where $|.|$ denotes the determinant. In practice, it is more realistic to assume that the unknown parameters belong to a user-specified parameter space Θ, which is comprised of all possible values for $\boldsymbol{\theta}$. Given Θ, we can find minimax optimal designs that minimize the maximum inefficiency over Θ and protect the experiment from the worst-case scenario over the parameter space. A minimax $D-$optimal design ξ^*_{min} is obtained by minimizing

$$\psi_{min}(\xi) = \max_{\boldsymbol{\theta} \in \Theta} -\log |M(\xi, \boldsymbol{\theta})|, \tag{5}$$

over the space of all designs on χ. The minimax problem (5) is a bi-level nested optimization problem with inner and outer optimization problems. Given any arbitrary design, the inner optimization problem is to maximize the $D-$criterion $-\log |M(\xi, \boldsymbol{\theta})|$ over Θ to find the maximum inefficiency and the outer optimization problem is to minimize the maximum of the inner problem over the space of all designs on χ. Alternatively, when a prior distribution $\pi_\Theta(\boldsymbol{\theta})$ is available for the unknown parameters on Θ, Bayesian optimal designs may also be found: a (pseudo) Bayesian $D-$optimal design ξ^*_{bayes} minimizes

$$\psi_{bayes}(\xi) = \int_{\boldsymbol{\theta} \in \Theta} -\log |M(\xi, \boldsymbol{\theta})| \pi_\Theta(\boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{6}$$

When $\pi_\Theta(\boldsymbol{\theta})$ is a discrete prior, the obtained designs are sometimes referred to as *optimum-on-average* or *robust* designs.

One advantage of working with approximate designs is existence of an equivalence theorem, which can be used to verify the optimality of a given design if the criterion is a convex function on the set of design measures. Each convex optimality criterion gives rise to a different equivalence theorem, but they generally have the same form. For example, a design ξ^*_{loc} is locally $D-$optimal if and only if the following inequality holds for all $\boldsymbol{x} \in \chi$,

$$c_{loc}(\boldsymbol{x}, \xi^*_{loc}) = \operatorname{tr} M^{-1}(\xi^*_{loc}, \boldsymbol{\theta}_0) I(\boldsymbol{x}, \boldsymbol{\theta}_0) - p \leq 0, \tag{7}$$

with equality in (7) at all support points of ξ^*_{loc}. The left hand-side of inequality (7) is sometimes called *sensitivity function*. The equivalence theorem for Bayesian $D-$optimality criterion is very similar (Kiefer and Wolfowitz 1959; Chaloner and Larntz 1989): a design ξ^*_{bayes} is a Bayesian $D-$optimal design if and only if the following inequality holds for all $\boldsymbol{x} \in \chi$,

$$c_{bayes}(\boldsymbol{x}, \xi^*_{bayes}) = \int_\Theta \operatorname{tr}\{M^{-1}(\xi^*_{bayes}, \boldsymbol{\theta}) I(\boldsymbol{x}, \boldsymbol{\theta})\} \pi(\boldsymbol{\theta}) d\theta - p \leq 0, \tag{8}$$

with equality in (8) at all support points of ξ^*_{bayes}. However, the equivalence theorem for a minimax type criterion takes on a more complicated form because (5) is not differentiable. The equivalence theorem states that a design ξ^*_{min} is minimax $D-$optimal among all the designs on χ if and only if there exists a probability measure μ^* on

$$A(\xi^*_{min}) = \left\{ \boldsymbol{\nu} \in \Theta \mid -\log |M(\xi^*_{min}, \boldsymbol{\nu})| = \max_{\boldsymbol{\theta} \in \Theta} -\log |M(\xi^*_{min}, \boldsymbol{\theta})| \right\}, \tag{9}$$

such that the following inequality holds for all $\boldsymbol{x} \in \chi$,

$$c_{min}(\boldsymbol{x}, \xi^*_{min}) = \int_{A(\xi^*_{min})} \operatorname{tr} M^{-1}(\xi^*_{min}, \boldsymbol{\nu}) I(\boldsymbol{x}, \boldsymbol{\nu}) \mu^* d(\boldsymbol{\nu}) - p \leq 0, \tag{10}$$

with equality in (10) at all support points of ξ^*_{min} (Wong 1992; Fedorov 1980; King and Wong 2000; Berger, King, and Wong 2000). The set $A(\xi^*_{min})$ is sometimes called the *answering set* of ξ^* and the measure μ^* is a sub-gradient of the non-differentiable criterion evaluated at $M(\xi^*_{min}, \nu)$. Understanding the properties of the sub-gradients and how to find them efficiently for the minimax optimal design problems present a key problem in solving this type of problems. In particular, there is no theoretical rule on how to choose the number of points in $A(\xi^*_{min})$ as support for the measure μ^* and they would have to be found by trial-and-error. For more details, see Masoudi *et al.* (2017). When χ is one or two dimensional, it is very common to plot the sensitivity function versus $\boldsymbol{x} \in \chi$ and visually inspect whether the graph meets the conditions in the equivalence theorem. If it does, the generated design is optimal; otherwise it is not optimal.

It is also possible to measure the efficiency of two arbitrary designs relative to each other. For example, given (4), we measure the proximity of a design ξ_1 to a design ξ_2 using its relative $D-$efficiency defined by

$$\text{eff}_{loc} = \left(\frac{|M(\xi_1, \boldsymbol{\theta})|}{|M(\xi_2, \boldsymbol{\theta})|} \right)^{1/p} = \exp\left(\frac{\psi_{loc}(\xi_2) - \psi_{loc}(\xi_1)}{p} \right), \tag{11}$$

where ξ_2 is usually a locally $D-$optimal design. The relative $D-$efficiency (11) may be interpreted in term of sample size; if its value is ρ, then ξ_1 requires $1/\rho$ times as many

observations to have the same $D-$efficiency as ξ_2. This means that, when ξ_2 is an optimal design, about $(1/\rho - 1)100\%$ more number of observations than for the optimal design will be needed to maintain the $D-$efficiency for ξ_1. Similarly, we can define Bayesian and minimax $D-$efficiencies by replacing ψ_{loc} with ψ_{min} and ψ_{bayes}, respectively.

When the design criterion is a convex functional, we can also use the equivalence theorem to quantify the proximity of a design to the optimal design without knowing the latter by means of the efficiency lower bound (ELB) defined by

$$\text{ELB} = \frac{p}{p + \max_{\boldsymbol{x}\in\chi} c(\boldsymbol{x}, \xi)}, \tag{12}$$

as a measure of proximity of a design to the optimum, where $c(\boldsymbol{x}, \xi)$ is one of the sensitivity functions defined above. According to the equivalence theorem, the value of (12) is between 0 and 1, and it is equal to 1 when the design is optimal. Atwood (1969) showed that (12) is the efficiency lower bound for the $D-$efficiency defined by (11). However, based on the equivalence theorem, we keep to use (12) as a measure of proximity to other types of optimal designs as well.

3. Imperialist Competitive Algorithm for Finding Optimal Designs

The imperialist competitive algorithm (ICA) is an evolutionary algorithm inspired from colonialism and socio-political behavior of humans, where developed countries attempt to take over or colonize less-developed countries to use their resources and extend their power (Atashpaz-Gargari and Lucas 2007). Within the optimization framework, ICA has a population of solutions called *countries*. In optimal design problems, each country is the location of the support points and the corresponding weights of a design on the space of all possible designs. ICA divides the population of countries into some sub-populations called *empires*. Each empire contains one *imperialist* and some *colonies*. The imperialist is the most powerful country within the empire. Here, the power of a country is defined to be a function of its cost value, i.e., criterion value. This means that, in a minimization problem, countries with smaller cost values are stronger. In ICA, there are two types of evolutionary moves: a) evolution within each empire, and, b) evolution among the empires. In the earlier, the colonies within each empire start to move or be absorbed toward their relevant imperialist country in a process called *assimilation* (Lin, Cho, and Chuan 2013). During this process, a colony may reach a better position than its imperialist. In this case, the imperialist loses its rank and the colony becomes the new imperialist. The assimilation improves searching around the better current solutions and so enhances the exploitation of the algorithm.

The evolution among the empires is achieved by a process called *imperialists competition*. In this process, the most powerful empires receive more chances to take possession of the colonies of the weakest empires. The competition step in ICA improves the exploration of the algorithm in a search for the global optimum. When an empire does not have any colony, it will be eliminated. ICA continues until it satisfies the stopping rule conditions. For more details, see Masoudi *et al.* (2017), Atashpaz-Gargari and Lucas (2007) and Hosseini and Al Khaled (2014).

To apply ICA for an optimal design problem, the user should first provide an initial guess about the number of support point $k(\geq p)$. In practice, the user can start by p and increment its value by one until the equivalence theorem confirms the optimality of the current best

design, which is the country with the least cost value. In optimal design problems, the ELB defined by (12) can be used to build a stopping rule condition for ICA. For example, the algorithm can be stopped when the value of the ELB of the best current design is larger than, say, 0.95. Clearly, finding ELB in each iteration increases the CPU time required by the algorithm as another optimization problem has to be solved to find maximum of the sensitivity function over χ. This is especially true for minimax and Bayesian type criteria, because the sensitivity function for the earlier involves solving a bi-level nested optimization problem and the latter requires approximating integrals. Therefore, we prefer to calculate the ELB periodically, say, after every 100 iterations, instead of every iteration to save the CPU time.

4. Implementation of Optimal Design Problems in ICAOD

Different functions are available to find optimal designs for nonlinear models in **ICAOD**: a) `locally()`: Finds locally optimal designs, b) `minimax()`: Finds minimax optimal designs , c) `bayes()`: Finds Bayesian optimal designs and d) `robust()`: Finds optimum-on-average or robust designs. Throughout this paper, we refer to them as "OD functions". **ICAOD** uses the S3 object oriented system and works with two objects of class '`minimax`' and '`bayes`'. Each class has its own `plot`, `print` and `update` method. The `plot` method is used to plot the sensitivity function and also calculate the ELB for the output design. The `update` method is for executing the algorithm for more number of iterations. For internal use, `locally()`, `minimax()` and `robust()` create an object of class '`minimax`', while `bayes()` works with an object of class '`bayes`'. For more details, see `?minimax` and `?bayes`. By default, OD functions are defined to determine $D-$optimal designs. In Section 6, we demonstrate how to specify user-defined optimality criteria. In what follows, the OD functions are explained in detail.

4.1. Locally Optimal Designs

The `locally()` function finds locally optimal designs and its main arguments are:

```
locally(formula, predvars, parvars, family = gaussian(), fimfunc = NULL,
        lx, ux, k, iter, ICA.control = list(), sens.control = list(),
        crt_func = NULL, sens_func = NULL,
        inipars)
```

The arguments in the first three lines of codes are common between the OD functions. Table 1 provides an overview of them. The arguments in the first line are required to construct the FIM of the model; `inipars` is equivalent to $\boldsymbol{\theta}_0$ in (4) and defines the vector of initial estimates for the model parameters.

The **ICAOD** package includes a formula interface to specify the model of interest. For example, assume the two-parameter logistic model defined by

$$f(x, \boldsymbol{\theta}) = \frac{1}{1 + \exp(-b(x - a))}, \tag{13}$$

where $\boldsymbol{\theta} = (a, b)$ is the vector of model parameters and x is the model predictor. To define (13) in **ICAOD**, we can set `formula = ~1/(1 + exp(-b * (x-a)))`, `predvars = "x"`, `parvars`

Argument	Description
`formula`	A formula that is the symbolic description of the nonlinear model.
`predvars`	A vector of characters that denote the model predictors in `formula`.
`parvars`	A vector of characters that denote the model parameters in `formula`.
`family`	The distribution of the model response and the link function. It is the same as the one in `glm()`. The default link function is `gaussian()`.
`fimfunc`	(optional) The Fisher information matrix (R `function`). Required if users wish to pass the FIM directly. It takes a `function` with arguments `x` (a vector of design points), `w` (a vector of associated weights) and `param` (a vector of model parameters). Only one of the `formula` and `fimfunc` arguments must be given.
`k`	The number of design points k.
`lx`	A vector of the lower bounds for the model predictors (design space χ).
`ux`	A vector of the upper bounds for the model predictors (design space χ).
`x`	(optional) A vector of design points $\boldsymbol{x}$. if given, only the optimal weights, `w`, are sought after. Required when the design points are pre-specified.
`ICA.control`	A `list` of ICA control parameters. By default, it will be created by `ICA.control()`. For more details, see Masoudi *et al.* (2017).
`iter`	The maximum number of iterations.
`sens.control`	Control Parameters of the maximization algorithm, which finds the maximum of the sensitivity function (7), (10) and (8) over the design space χ. The obtained maximum is used to calculate the ELB of a design. By default, it will be created by `sens.control()`.
`crt_func`	(optional) A user-specified criterion (R `function`).
`sens_func`	(optional) A user-specified sensitivity function (R `function`).

Table 1: Overview of the most important common arguments of the OD functions.

`= c("a", "b")` and `family = "binomial"` (or `family = binomial()`). Alternatively, one may pass the FIM of (13) as an R `function` via the argument `fimfunc` directly. In this option, the arguments of the defined function must be a) `x`: is a `vector` of $(\boldsymbol{x}_1, ..., \boldsymbol{x}_k)$ in (2), b) `w`: is a `vector` of $(w_1, ..., w_k)$ in (2), and c) `param`: is a `vector` of $\boldsymbol{\theta}$ in (13). The output is the FIM of (13) evaluated at the given `x`, `w` and `param` as a `matrix`.

The argument `sens.control` is a list of control parameters for `nloptr()` available in the **nloptr** package (Johnson 2014). This function is used here to solve $\max_{\boldsymbol{x}\in\chi} c(\boldsymbol{x}, \xi)$ for computing the ELB (12). When not given, it will be created automatically by the function `sens.control`. We recommend not to change its default values as they have been successfully tested for a large number of problems.

The `crt_func` and `sens_func` arguments are used to find a user-defined optimal designs, which are described in Section 6.

4.2. Minimax Optimal Designs

The `minimax()` function finds minimax optimal designs and its main arguments are:

```
minimax(formula, predvars, parvars, family = gaussian(), fimfunc = NULL,
```

```
lx, ux, k, iter, ICA.control = list(), sens.control = list(),
crt_func = NULL, sens_func = NULL,
lp, up, n.grid = 0,
sens.minimax.control = list(), crt.minimax.control = list())
```

The first three lines of codes are similar to the ones in `locally()` and the rest of the arguments are used to evaluate the minimax criterion (5) and its sensitivity function (10) at a given design. Table 2 presents an overview of the arguments specifically available in `minimax()`.

In **ICAOD**, the parameter space Θ are either continuous or discrete. Note that, the lower bound and upper bound of Θ are specified via the arguments `lp` and `up`, respectively. When Θ is continuous, **ICAOD** uses `nloptr()` to solve the inner maximization problem in (5) over Θ at a given design. The default optimization algorithm from `nloptr()` is the DIRECT-L algorithm, which is a deterministic search algorithm based on the systematic division of the search domain into smaller and smaller hyperrectangles (Gablonsky and Kelley 2001). For our applications, the most influential tuning parameter of `nloptr()` is the maximum number of function evaluations denoted by `maxeval` (its default value is 1000) via the `crt.minimax.control` argument. The parameter space may also be discretized. In this option, the total number of grid points is equal to `n.grid^p`. When specified, ICA evaluates the criterion at these grid points to solve the maximization problem over Θ.

4.3. Bayesian Optimal Designs

The `bayes()` function finds Bayesian optimal designs and its main arguments are:

```
bayes(formula, predvars, parvars, family = gaussian(), fimfunc = NULL,
      lx, ux, k, iter, ICA.control = list(), sens.control = list(),
      crt_func = NULL, sens_func = NULL,
      prior, crt.bayes.control = list(), sens.bayes.control = list())
```

The first three lines of codes are similar to the ones in `locally()` and the rest of the arguments are used to approximate the integrals in (6) and (8) at a given design. Table 2 presents an overview of the arguments specifically available in `bayes()`.

By default, **ICAOD** uses the `hcubature()` function from the **cubature** package (Johnson 2013; Narasimhan and Johnson 2017) to approximate the integrals. Th function `hcubature()` includes an adaptive multidimensional integration method over hypercubes known as hcubature algorithm (Berntsen, Espelid, and Genz 1991; Genz and Malik 1980). For our applications, the most important tuning parameters of the hcubature algorithm are the maximum number of integrand evaluations `maxEval` (its default value is `50000`) and a user-specified tolerance `tol` (its default value is `1e-5`). This algorithm stops either when the integral error estimate is less than the integral estimate multiplied by its value or when the it reaches the specified maximum number of function evaluations `maxEval`, whichever happens earlier. When the prior distribution is less diffuse, it is sometimes more efficient to reduce the value of `maxEval` to increase the speed of the hcubature algorithm.The control parameters of the `hcubature()` function can be regulated via the argument `crt.bayes.control`.

Alternatively, **ICAOD** also offers the Gauss-Legendre and the Gauss-Hermite formulas to approximate the integrals. These methods are implemented in ICA using the **mvQuad** package

Argument	function	Description
`lp`	`minimax()`	A vector of lower bounds for $\boldsymbol{\theta}$.
`up`		A vector of upper bounds for $\boldsymbol{\theta}$.
`n.grid`		(optional) When have a positive value, the parameters space Θ will be discretized, where the number of grid points will be equal to `n.grid^p` (defaults to 0).
`crt.minimax.control`		A list of control parameters of the function `nloptr()`, which is used to maximize the optimality criterion at a given design over Θ. By default, it will be created by `crt.minimax.control()`.
`sens.minimax.control`		A list of control parameters to find the answering set (9), which is required to obtain the sensitivity function and calculate the ELB. By default, it will be created by `sens.minimax.control()`. For more details, see `?sens.minimax.control`.
`prior`	`bayes()`	An object of class '`cprior`' that contains the necessary information about the prior distribution for the unknown parameters $\boldsymbol{\theta}$. For popular prior distributions, it can be created via the `uniform()`, `normal()`, `skewnormal()`, `student()` functions. For more details, see `?bayes`.
`crt.bayes.control`		A list of control parameters to approximate the integrals in (6), using either the `hcubature()` function (an adaptive multidimensional integration method over hypercubes) or the Gaussian quadrature formulas implemented by the **mvQuad** package. By default, it will be created by `crt.bayes.control()`.
`sens.bayes.control`		A list of control parameters required to approximate the integrals in (8). It is very similar to `crt.bayes.control()` and by default will be created by `crt.bayes.control()`.
`prob`	`robust()`	A `vector` of the probability measure associated with each vector of initial estimates for the unknown parameters $\boldsymbol{\theta}$.
`parset`		A `matrix` where each of its row is a vector of the initial estimates for $\boldsymbol{\theta}$.

Table 2: Overview of the arguments that are used to evaluate minimax, Bayesian and robust (optimum-on-average) optimality criteria at a given design.

(Weiser 2016) and can be requested via the argument `crt.bayes.control`. For more details, see `?mvQuad::createNIGrid()`.

4.4. Robust or Optimum-On-Average Designs

The `robust()` function finds optimum-on-average or robust designs and its main arguments are:

```
robust(formula, predvars, parvars, family = gaussian(), fimfunc = NULL,
       lx, ux, k, iter, ICA.control = list(), sens.control = list(),
       crt_func = NULL, sens_func = NULL,
       prob, parset)
```

The first three lines of codes are similar to the ones in `locally()` and the rest of the arguments are used to evaluate the optimum-on-average criterion at a given design. Table 2 presents an overview of the arguments specifically available in `robust()`.

5. Examples

In this section, we provide two examples to show the functionality of the **ICAOD** package to determine optimal designs. In the first example, we find locally and minimax $D-$optimal designs for a logistic model with applications in educational testing. In the second example, we specify Bayesian and robust optimal designs for the sigmoid Emax model with applications in dose-response studies.

5.1. Logistic Model with A Single Predictor

The logistic model is very popular for modeling binary outcomes. For example, consider an educational research that studies the effect of hours of practice on the mastery of a mathematical task. Let Y be a binary response variable that takes the value 1 if a subject has mastered the task and 0 otherwise. The logistic model is defined by

$$f(x, \boldsymbol{\theta}) = P(Y = 1) = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)}, \tag{14}$$

where x is the hours of practice and $\boldsymbol{\theta} = (\beta_0, \beta_1)^T$. Assume that for each subject up to six hours of practice are possible, i.e., $x \in \chi = [0, 6]$. If the purpose of the study is to estimate the model parameters accurately, an appropriate criterion is the $D-$optimality. The design questions here are a) what is the best number of levels of x to apply in the study, b) what are these levels and c) how many subjects should be assigned to each level? For example, a researcher may choose a uniform design that includes an equal number of subjects who have practiced for $0, 1, 2, 3, 4, 5, 6$ hours. We denote this design by

$$\xi_{uni} = \left\{ \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 & 1/7 \end{array} \right\}. \tag{15}$$

The FIM of model (14) depends on the unknown parameters through $\frac{\partial f(\boldsymbol{x}, \boldsymbol{\theta})}{\partial \beta_j}, j = 0, 1$. Following Berger and Wong (2005), let $\boldsymbol{\theta}_0 = (-4, 1.3333)^T$ be the best initial guess for $\boldsymbol{\theta}$ available from, say, a similar study. In **ICAOD**, the locally $D-$optimal design is found by

```
R> library("ICAOD")
R> log1 <- locally(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+                  predvars = "x", parvars = c("b0", "b1"),
+                  family = "binomial", lx = 0, ux = 6, iter = 40, k = 2,
+                  inipars = c(-4, 1.3333), ICA.control = list(rseed = 1))
R> print(log1)

**********************************************************************
ICA iter: 40
 Points1 Points2
 1.84249 4.15765
 Weights1 Weights2
  0.500   0.500
Criterion value:  3.568679
Convergence: Maximum_Iteration
CPU time: 3.159  seconds!
**********************************************************************
```

Throughout this paper, the `rseed` argument is used to guarantee the reproducibility of the results. The algorithm stopped at iteration number 40 because it reached the maximum number of iterations (`iter = 40`). Here, the design provided by the output assigns equal weights to $x_1 = 1.84249$ and 4.15765. This mean that, half of the subjects should be assigned to practice nearly less than 2 hours and the other half should practice a little bit more than 4 hours. The $D-$criterion (4) evaluated at this design is equal to 3.5686. The plot of the sensitivity function of the design provided by the output and the value of the ELB is obtained by

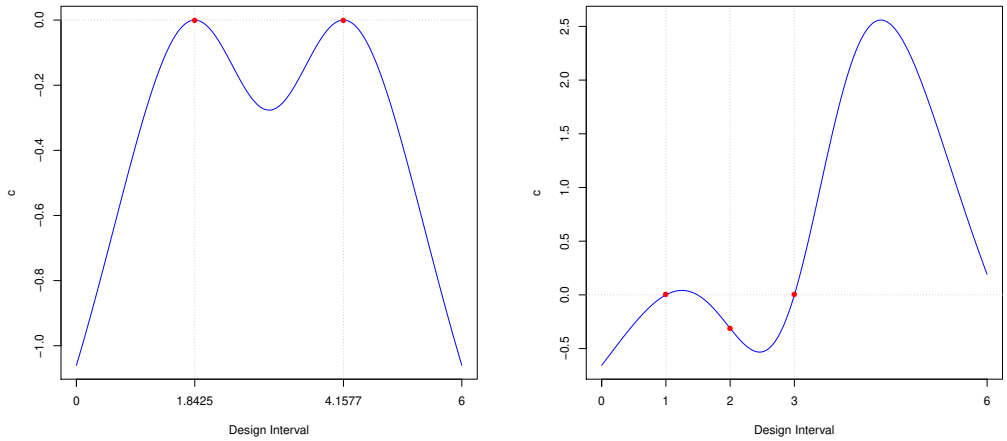
```
R> plot(log1)

**********************************************************************
Maximum of the sensitivity function is  8.543706e-07
Efficiency lower bound (ELB) is  0.9999996
Verification required 0.269 seconds!
**********************************************************************
```

Figure 1 (a) displays the plot of the sensitivity function (7) of the design provided by the output on the design space $[0, 6]$. Based on the equivalence theorem, this design is optimal because the sensitivity function is equal or less than zero on $[0, 6]$ and (roughly) equal to zero at 1.84249 and 4.15765 (see the red points). The value of the ELB is nearly 1, which also indicates the optimality of this design.

It is interesting to assess the performance of the uniform design ξ_{uni} with respect to the locally $D-$optimal design obtained above. Using (11), we can calculate the $D-$efficiency of ξ_{uni} relative to the locally $D-$optimal design by

```
R> leff(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+      predvars = "x", parvars = c("b0", "b1"),
+      family = "binomial", inipars = c(-4, 1.3333),
```



(a) `Output from plot(log1)` (b) `Output from plot(log3)`

Figure 1: Plots of the sensitivity functions of the designs generated by the `locally()` function for the logistic model over $\chi = [0, 6]$ when $\boldsymbol{\theta} = \boldsymbol{\theta}_0 = (-4, 1.3333)^T$.

```
+       x1 = c(0:6), w1 = rep(1/7, 7),
+       x2 = log2$evol[[20]]$x, w2 = log2$evol[[20]]$w)

[1] 0.7778723
```

The value of the relative $D-$efficiency indicates that ξ_{uni} requires about $100(1/0.777 - 1) = 29\%$ more number of subjects to have the same $D-$efficiency as the $D-$optimal design when $\boldsymbol{\theta} = \boldsymbol{\theta}_0$. Therefore, having subjects to practice, say, less than 1 hours or more than 5 hours will not increase the efficiency of the parameter estimates very much.

The value of the ELB may also be used to construct a stopping rule condition for ICA. This feature is activated via the `ICA.control` argument in all OD functions similar to what follows.

```
R> log2 <- locally(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+                  predvars = "x", parvars = c("b0", "b1"),
+                  family = "binomial", lx = 0, ux = 6, iter = 40, k = 2,
+                  inipars = c(-4, 1.3333),
+                  ICA.control = list(rseed = 1,
+                                     checkfreq = 20,
+                                     stop_rule = "equivalence",
+                                     stoptol = .99))
R> print(log2)

**********************************************************************
ICA iter: 20
 Points1 Points2
 1.84420 4.15857
 Weights1 Weights2
```

```
  0.500   0.500
Criterion value:  3.56868
Convergence: equivalence
CPU time: 2.053  seconds!
*****************************************************************
*****************************************************************
Maximum of the sensitivity function is  8.070843e-05
Efficiency lower bound (ELB) is  0.9999596
Verification required 0.367 seconds!
*****************************************************************
```

We set `stop_rule = "equivalence"` to activate the stopping rule that is based on the equivalence theorem. In this case, ICA starts to calculate the ELB for the best design every `checkfreq = 20` iterations and it stops whenever the value of the ELB is larger than `stoptol = .99`. In this example, ICA stopped at the first check run because the value of ELB is 0.999($>$ `stoptol`). Note that, we requested to calculate the ELB after every 20 iterations, instead of every iteration, to prevent a significant increase in the CPU time. This equivalence-based stopping rule is also available in other OD functions. However, we note that, optimality verification for Bayesian or minimax type criteria is more complicated and may slow down the ICA.

ICAOD can also handle a situation where the design points are pre-specified, but their optimal associated weights are of interest. For example, assume that the experimental resources only allow a pre-specified hours of practice, say, $x_1 = 1$, $x_2 = 2$, $x_3 = 3$ hours. In all OD functions, the design points can be specified similarly via the argument `x` (a vector of design points):

```
R> log3 <- locally(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+                  predvars = "x", parvars = c("b0", "b1"),
+                  family = "binomial", lx = 0, ux = 6, iter = 40,
+                  x = c(1, 2, 3),
+                  inipars = c(-4, 1.3333),
+                  ICA.control = list(rseed = 1, checkfreq = Inf))

R> print(log3)


*****************************************************************
ICA iter: 40
 Weights: 0.500 0.000 0.500
Criterion value:  4.187342
Convergence: Maximum_Iteration
CPU time: 3.3  seconds!
*****************************************************************
*****************************************************************
Maximum of the sensitivity function is  2.558775
Efficiency lower bound (ELB) is  0.4387144
Verification required 0.305 seconds!
*****************************************************************
```

The results show that no weight should be assigned to the subjects with 2 hours of practice. This means that, the responses from subjects with 2 hours of practice will not increase the efficiency of estimation very much. Hence, this level may be eliminated to save more resources.

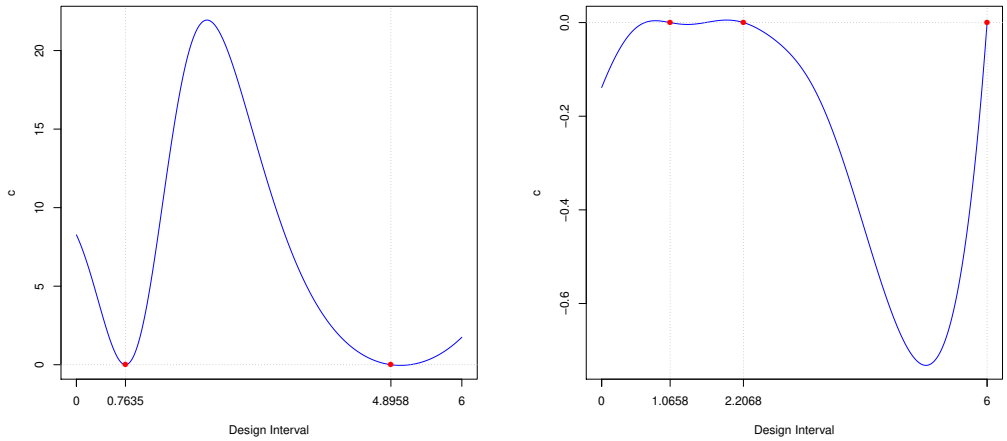
The value of the ELB and the plot of the sensitivity function in Figure 1 (b) clearly show that the obtained design is not globally optimal. This comes as no surprise because the given design points in `x` do not belong to the support of the optimal design when $\boldsymbol{\theta} = \boldsymbol{\theta}_0$. Note that, `checkfreq = Inf` requests a `plot` method for the design provided by the output so that `plot()` is not required anymore. For space consideration, we use this option in the rest of this paper.

Locally optimal designs usually lose their efficiency when the parameter estimates are far from their true unknown values. Moreover, in practice, it is more realistic to assume that the parameters belong to a parameter space, rather than fixing their values at some points. For example, let $\boldsymbol{\theta} = (\beta_0, \beta_1)^T$ belongs to $\Theta = [\beta_0^L, \beta_0^U] \times [\beta_1^L, \beta_1^U]$, where $\beta_0^L = -6$, $\beta_0^U = -2$, $\beta_1^L = .5$ and $\beta_1^U = 2$. As a conservative strategy, a minimax $D-$optimal design minimizes the maximum inefficiency over Θ. To find the minimax $D-$optimal design for our design setting, we first set $k = 2$ to find the minimax $D-$optimal design within the class of two-point designs:

```
R> log4 <- minimax(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+                  predvars = "x", parvars = c("b0", "b1"),
+                  family = "binomial",
+                  lx = 0, ux = 6, lp = c(-6, .5), up = c(-2, 2),
+                  iter = 200, k = 2,
+                  ICA.control = list(rseed = 1,
+                                     checkfreq = 50,
+                                     stop_rule = "equivalence",
+                                     stoptol = .99),
+                  crt.minimax.control = list(optslist = list(maxeval = 200)))
R> print(log4)

**********************************************************************
ICA iter: 200
 Points1 Points2
 0.76347 4.89579
 Weights1 Weights2
  0.500   0.500
Criterion value:  7.782754
Convergence: Maximum_Iteration
CPU time: 180.64  seconds!
**********************************************************************
**********************************************************************
Maximum of the sensitivity function is  21.94145
Efficiency lower bound (ELB) is  0.08353714
Verification required 0.623 seconds!
Adjust the value of 'n_seg' in 'sens.minimax.control' for higher speed.
**********************************************************************
```

To increase the CPU time, we reduced the value of `maxeval` from 1000 (default value) to 200.



(a) $k = 2$ (b) $k = 3$

Figure 2: Plots of the sensitivity functions of the two- and three-point designs generated by the `minimax()` function for the logistic regression model over $\chi = [0, 6]$ when $\Theta = [-6, -2] \times [0.5, 2]$. The plot (b) shows the nearly optimality of the three-point design.

Figure 2 (a) displays the sensitivity plot of the design by provided by the output and it does not verify the optimality of the two-point design. Therefore, we increment the value of k by one and re-execute the above code:

```
R> log5 <- minimax(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+                  predvars = "x", parvars = c("b0", "b1"),
+                  family = "binomial",
+                  lx = 0, ux = 6, lp = c(-6, .5), up = c(-2, 2),
+                  iter = 500, k = 3,
+                  ICA.control = list(rseed = 1,
+                                     checkfreq = 50,
+                                     stop_rule = "equivalence",
+                                     stoptol = .99),
+                  crt.minimax.control = list(optslist = list(maxeval = 200)))
R> print(log5)

**************************************************************************
ICA iter: 200
 Points1 Points2 Points3
 1.06581 2.20682 6.00000
 Weights1 Weights2 Weights3
  0.124   0.383   0.493
Criterion value:  6.736316
Convergence: equivalence
CPU time: 194.165  seconds!
**************************************************************************
```

```
**********************************************************************
Maximum of the sensitivity function is  0.005312662
Efficiency lower bound (ELB) is  0.9973507
Verification required 1.516 seconds!
Adjust the value of 'n_seg' in 'sens.minimax.control' for higher speed.
**********************************************************************
```

Figure 2 (b) displays the plot of the sensitivity function of the three-point generated design and it indicates its nearly optimality. The optimal design suggests subjects with nearly 1, 2 and 6 hours of practice, where roughly half of the subjects should be assigned to practice for 6 hours.

Similar to the locally $D-$optimal design, we can assess the minimax $D-$efficiency of ξ_{uni} with respect to the minimax $D-$optimal design by

```
R> meff(formula = ~exp(b0 + b1 * x)/(1 + exp(b0 + b1 * x)),
+       predvars = "x", parvars = c("b0", "b1"),
+       family = "binomial",
+       lp = c(-6, .5), up = c(-2, 2),
+       x1 = c(0:6), w1 = rep(1/7, 7),
+       x2 = log5$evol[[200]]$x, w2 = log5$evol[[200]]$w)

[1] 0.74089
```

This value indicates that ξ_{uni} requires about $100(1/0.74089 - 1) = 35\%$ more subjects to have the same minimax $D-$efficiency as the minimax $D-$optimal design when $\Theta = [-6, -2] \times [0.5, 2]$.

5.2. Sigmoid-Emax Model

The sigmoid Emax model is commonly used in pharmacokinetics/pharamacodynamics to describe the S-shape dose-response relationship (see, e.g, Macdougall 2006; Thomas 2006). This model is defined by

$$E(Y) = f(x, \boldsymbol{\theta}) = \beta_1 + (\beta_2 - \beta_1)\frac{x^{\beta_4}}{x^{\beta_4} + \beta_3^{\beta_4}}, \tag{16}$$

where x is the dose level (in mg), $x \in \chi = (0, x_0]$, x_0 is user-selected and $\boldsymbol{\theta} = (\beta_1, \beta_2, \beta_3, \beta_4)^T$, $\theta_2 > \beta_1$, $\beta_3 > 0$. All errors are assumed to be independent and normally distributed with mean zero and constant variance. Here, β_1 is the minimum mean response, β_2 is the maximum mean response, β_3 is the ED50, i.e., the dose at which 50 percent of the maximum mean effect is achieved, and β_4 is the slope parameter.

In dose-response studies, optimal designs usually determine how many doses are required to be tested, what are their levels, and how many subjects to allocate to each dose level. Let $\chi = (0, 1000]$mg. Similar to Dragalin, Hsuan, and Padmanabhan (2007) and Wang and Yang (2014), we are interested in efficient estimation of $\boldsymbol{\theta}$ and the $D-$optimality is an appropriate design criterion for this purpose.

Using some algebra, it is straightforward to show that the FIM of the sigmoid Emax model depends on the unknown parameters $\boldsymbol{\theta}$. This parameter dependency must be dealt with based on the type of information available on $\boldsymbol{\theta}$. For example, using information from a pilot study, one may elicit a uniform prior distribution for $\boldsymbol{\theta}$ and search for Bayesian optimal designs. As an illustrative example, let $\beta_1 \sim U(4,8)$, $\beta_2 \sim U(11,15)$, $\beta_3 \sim U(100,130)$ and $\beta_4 \sim U(5,9)$, and all the uniform prior distributions be independent. For simplicity, we denote the independent uniform distributions for $\beta_i, i = 1,2,3,4$ by π_Θ, where $\Theta = [4,8] \times [11,15] \times [100,130] \times [5,9]$ is the parameter space. This prior can be defined in **ICAOD** by the `uniform()` function as follows.

```
R> prior1 <- uniform(lower = c(4, 11, 100, 5), upper = c(8, 15, 130, 9))
```

Here, the output is an object of class '`cprior`', which can be passed to the argument `prior` of the `bayes()` function.

To find the number of support points for the Bayesian $D-$optimal design, we repeated the same incremental process as described in Section 5.1 for finding minimax optimal design. This process is excluded here due to space consideration. The Bayesian $D-$optimal design has 5 points in its support, which are found by

```
R> sig1 <- bayes(formula = ~b1 + (b2-b1) * x^b4/(x^b4 + b3^b4),
+                predvars = "x",
+                parvars = c("b1", "b2", "b3", "b4"),
+                lx = .001, ux = 1000, k = 5, iter = 400, prior = prior1,
+                ICA.control = list(rseed = 1, checkfreq = Inf))
R> print(sig1)

**********************************************************************
ICA iter: 400
  Points1   Points2   Points3   Points4   Points5
 0.17040   94.59828  113.69179 138.35282 999.99946
 Weights1  Weights2  Weights3  Weights4  Weights5
   0.243     0.194     0.116     0.203     0.244
Criterion value:  12.72082
Convergence: maxiter
CPU time: 391.99  seconds!
**********************************************************************
**********************************************************************
Maximum of the sensitivity function is  0.0001633976
Efficiency lower bound (ELB) is  0.9999592
Verification required 33.871 seconds!
Adjust the control parameters in 'sens.bayes.control' for higher speed
**********************************************************************
```

Figure 3 (a) presents the plot of the sensitivity function of the five-point design provided by the output and it verifies its optimality. In our example, the Bayesian $D-$optimal design suggests five dose levels, with four of them located below 140mg and one located at the maximum. Roughly 50% of the observations should be assigned to the lower and upper bound of the

dose interval. Note that, the result can also be obtained in lesser CPU time if we adjust the control parameters of the integral approximations via the argument `crt.bayes.control`. For a discussion on these tuning parameters, see Masoudi, Holling, Duarte, and Wong (2019).

Using a non-optimal design may be inefficient even when its design points are sampled uniformly from the design space. As an illustrative example, assume a situation where a researcher decides to work with an equally-weighted uniform design that has 11 points located on $0.001, 100, 200, 300, ..., 1000$. This design is not optimal when $\boldsymbol{\theta} \sim \pi_\Theta$. The Bayesian $D-$efficiency of the uniform design with respect to the obtained Bayesian $D-$optimal design is calculated by

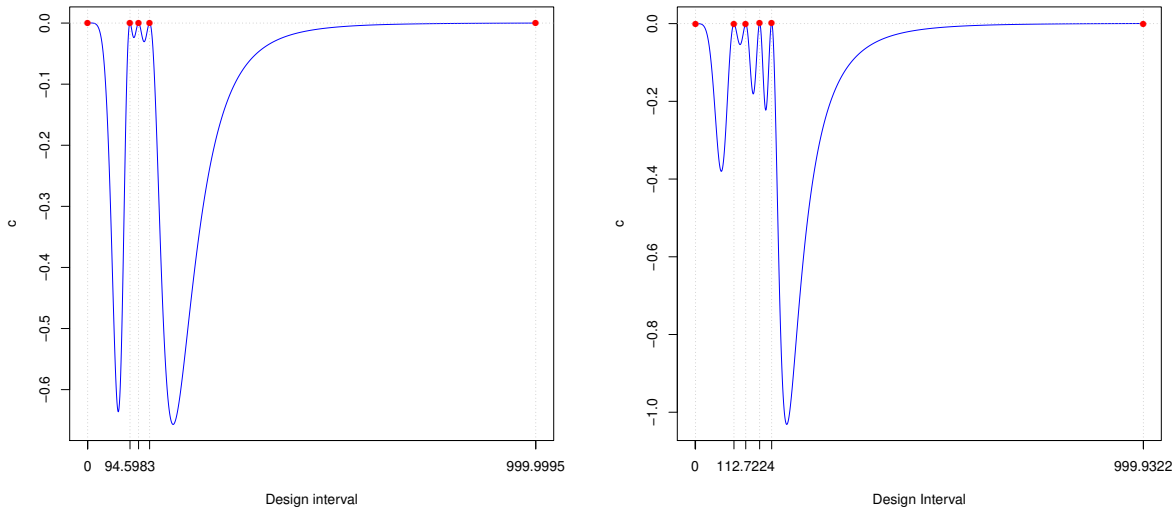
```
R> beff(formula = ~b1 + (b2-b1) * x ^b4/(x^b4 + b3^b4),
+      predvars = "x",
+      parvars = c("b1", "b2", "b3", "b4"),
+      prior = prior1,
+      x1 = c(.001,seq(100, 1000, by = 100)),
+      w1 = rep(1/11, 11),
+      x2 = sig1$evol[[400]]$x, w2 = sig1$evol[[400]]$w)

[1] 0.3063289
```

The non-optimal design may seem to have fairly chosen, but its Bayesian $D-$efficiency value suggests that, roughly 226% more observations are needed to maintain the $D-$efficiency for the non-optimal design in comparison to the Bayesian $D-$optimal design when $\boldsymbol{\theta} \sim \pi_\Theta$. The `bayes()` function is very flexible and can incorporate different prior distributions. For more details, see Masoudi *et al.* (2019).

ICAOD can also find robust or optimum-on-average designs when the prior distributions are discrete. As an illustrative example, assume $\Theta_0 = \{\boldsymbol{\theta}_{01}, \boldsymbol{\theta}_{02}, \boldsymbol{\theta}_{03}, \boldsymbol{\theta}_{04}, \boldsymbol{\theta}_{05}\}$ be a set of five vectors of initial estimates for $\boldsymbol{\theta} = (\beta_1, \beta_2, \beta_3, \beta_4)$, where $\boldsymbol{\theta}_{01} = (4, 11, 100, 5)$, $\boldsymbol{\theta}_{02} = (5, 12, 110, 6)$, $\boldsymbol{\theta}_{03} = (6, 13, 120, 7)$, $\boldsymbol{\theta}_{04} = (8, 15, 130, 9)$ and $\boldsymbol{\theta}_{05} = (12, 30, 160, 13)$. Let π_{Θ_0} denotes a discrete uniform prior distribution that assigns the same probability to each vector element of Θ_0. The six-point optimum-on-average design is given by

```
R> parset1 <- matrix(c(4, 11, 100, 5,
+                      5, 12, 110, 6,
+                      6, 13, 120, 7,
+                      8, 15, 130, 9,
+                      12, 30, 160, 13),
+                    nrow = 5, byrow = TRUE)
R> sig2 <- robust(formula = ~b1 + (b2-b1) * x ^b4/(x^b4 + b3^b4),
+                 predvars = "x",
+                 parvars = c("b1", "b2", "b3", "b4"),
+                 lx = .001, ux = 1000, k = 6, iter = 400,
+                 parset = parset1,
+                 prob = rep(1/5, 5),
+                 ICA.control = list(rseed = 1, checkfreq = Inf))
R> print(sig2)
```



(a) $\boldsymbol{\theta} \sim \pi_{\Theta}$ (b) $\boldsymbol{\theta} \sim \pi_{\Theta_0}$

Figure 3: The plots of sensitivity functions of the generated designs for the sigmoid Emax model over the design space $[0.001, 1000]$. The left panel displays the plot of the sensitivity function of the design generated by the function `bayes()` when $\boldsymbol{\theta} \sim \pi_{\Theta}$. The right panel displays the plot of the sensitivity function of the design generated by the function `robust()` when $\boldsymbol{\theta} \sim \pi_{\Theta_0}$.

```
**************************************************************************
ICA iter: 400
  Points1   Points2   Points3   Points4   Points5   Points6
 0.73419   86.42749  112.72244 143.73056 170.57625 999.93217
 Weights1  Weights2  Weights3  Weights4  Weights5  Weights6
    0.200     0.132     0.155     0.186     0.098     0.229
Criterion value:  12.21398
Convergence: Maximum_Iteration
CPU time: 68.501  seconds!
**************************************************************************
**************************************************************************
Maximum of the sensitivity function is  0.0002345949
Efficiency lower bound (ELB) is  0.9999414
Verification required 1.462 seconds!
**************************************************************************
```

Figure 3 (b) displays the plot of the sensitivity function of the design provided by the output and it verifies the optimality of the six-point design. Similar to the optimal design generated by `bayes()`, the generated design here allocates most of its support points to the lower half of the dose interval.

6. User-Specified Optimality Criteria

ICAOD can also find optimal designs with respect to user-specified optimality criteria. In this section, as an illustrative example, we find $c-$optimal designs for the two-parameter logistic (2PL) model with applications in dose-response studies. The 2PL model is commonly used in dose-response studies to model the relationship between the dose level of a drug and the probability of a success, e.g, the probability that patients are cured. This model is defined by

$$f(x, \boldsymbol{\theta}) = P(Y = 1) = \frac{1}{1 + \exp(-b(x - a))}, \tag{17}$$

where x is the dose level (predictor), $\boldsymbol{\theta} = (a, b)^T$, b is the slope parameter and a is the dose level at which the response probability is 0.5 (ED50). Throughout this paper, we denote the dose level at which the response probability is equal to π by ED100π. For the 2PL model, it can be shown that ED100π is equal to $c(\boldsymbol{\theta}) = a + \gamma b^{-1}$, where $\gamma = \log[\pi/(1 - \pi)]$ (see, for example, Zhu and Wong 2001).

Sometimes the purpose of a study is to estimate a function of the unknown parameters, say, ED100π, rather than estimating all the parameters simultaneously. For example, in heart defibrillator design problems, estimating the ED95, or equivalently, estimating $c(\boldsymbol{\theta}) = a + \log(0.95/(1 - 0.95))b^{-1}$ for the 2PL model is of interest (Clyde, Müller, and Parmigiani 1995). In this case, a reasonable optimality criterion is the one that minimizes the asymptotic variance of the maximum likelihood (ML) estimator of $c(\boldsymbol{\theta})$, which is proportional to

$$\psi^c(\xi, \boldsymbol{\theta}) = \nabla^T c(\boldsymbol{\theta}) M^{-1}(\xi, \boldsymbol{\theta}) \nabla c(\boldsymbol{\theta}), \tag{18}$$

where $\nabla c(\boldsymbol{\theta})$ is the gradient of $c(\boldsymbol{\theta})$ and $M^{-1}(\xi, \boldsymbol{\theta})$ is the inverse of the FIM (see, e.g, Silvey 1980, page 4). For the 2PL model, $\nabla c(\boldsymbol{\theta}) = (1, -\gamma b^{-2})^T$. In the optimal design literature, $\psi^c(\xi, \boldsymbol{\theta})$ is referred to as $c-$optimality criterion and a design that minimizes $\psi^c(\xi, \boldsymbol{\theta})$ is called $c-$optimal design. An equivalence theorem is also available for $c-$optimality: a design ξ_c^* is $c-$optimal among all the designs on χ if and only if the following inequality holds for all $x \in \chi$,

$$c^c(x, \xi_c^*) = \text{tr}(B(\boldsymbol{\theta}) M^{-1}(\xi, \boldsymbol{\theta}) M(\xi_x, \boldsymbol{\theta}) M^{-1}(\xi, \boldsymbol{\theta})) - \psi^c(\xi, \boldsymbol{\theta}) \leq 0, \tag{19}$$

with equality in (19) for all the support points of ξ_c^* (see, e.g, Chaloner and Larntz 1989). Here, $B(\boldsymbol{\theta}) = \nabla^T c(\boldsymbol{\theta}) \nabla c(\boldsymbol{\theta})$ and ξ_x denotes a degenerate design that puts all its mass on x.

Similar to the $D-$optimality criterion, $c-$optimality also depends on the unknown parameters and different types of optimal designs may be found, depending on how to deal with the unknown parameters. As benchmark examples, in this section, we find locally and Bayesian $c-$optimal designs for estimating the ED95 for the 2PL model when $\chi = [-1, 1]$. These examples are also available in Chaloner and Larntz (1989). Finding a minimax $c-$optimal or a robust design is very similar and is excluded due to space consideration.

To use **ICAOD** for finding $c-$optimal designs, the user should first define the $c-$optimality criterion and its sensitivity function as two separate functions in the R environment. Later, these functions will be passed to `bayes()`, `minimax()`, `locally()` and `robust()` via the `crtfunc` and `sensfunc` arguments, respectively. For example, given the 2PL model with parameters `parvars = c("a", "b")`, the following lines of codes define (18) and (19) in the R environment to be used in `locally()`, `minimax()` and `robust()`.

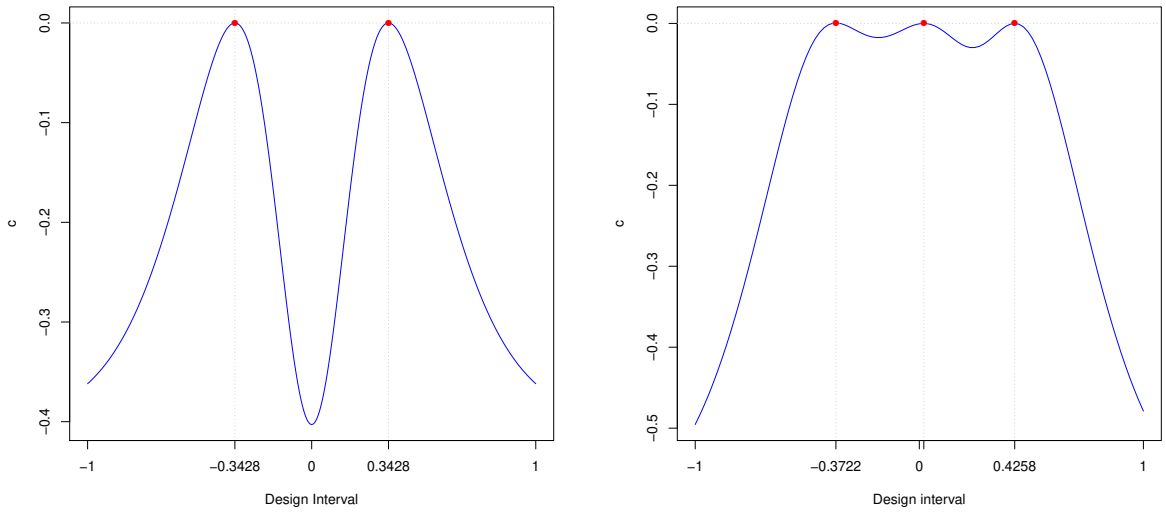
```
R> c_opt <-function(x, w, a, b, fimfunc){
+   gam <- log(.95/(1-.95))
+   M <- fimfunc(x = x, w = w, a = a, b = b)
+   c <- matrix(c(1, -gam * b^(-2)), nrow = 1)
+   B <- t(c) %*% c
+   sum(diag(B %*% solve(M)))
+ }

R> c_sens <- function(xi_x, x, w, a, b, fimfunc){
+   gam <- log(.95/(1-.95))
+   M <- fimfunc(x = x, w = w, a = a, b = b)
+   M_inv <- solve(M)
+   M_x <- fimfunc(x = xi_x, w = 1, a = a, b = b)
+   c <- matrix(c(1, -gam * b^(-2)), nrow = 1)
+   B <- t(c) %*% c
+   sum(diag(B %*% M_inv %*% M_x %*%  M_inv)) - sum(diag(B %*% M_inv))
+ }
```

The arguments `x`, `w` are, respectively, the vector of design points and their associated weights defined by (2). `fimfunc()` is a function with arguments `x`, `w`, `a` and `b` that returns the evaluated FIM as a `matrix` and `xi_x` denotes a degenerate design, which has the same length as the number of model predictors. The arguments `a` and `b` are model-specific and denote the parameters of the model that is specified via `parvars`. A convenient feature of **ICAOD** is that there is no need to compute the FIM of the model even for a user-specified optimality criterion and the user can apply the internally-created FIM within the body of `c_opt()` and `c_sens()` using `fimfunc()`. Note that, both of the `c_opt()` and `c_sens()` functions are not vectorized with respect to `a` and `b`. This means that `fimfunc()` returns only a `matrix`, and `c_opt()` and `c_sens()` return a value. This is a necessary structure required by the `locally()`, `minimax()` and `robust()` functions. The following lines of codes provide the locally $c-$optimal design for estimating the ED95 when $\boldsymbol{\theta} = (0, 7)$.

```
R> twoPL1 <- locally(formula = ~1/(1 + exp(-b * (x-a))), predvars = "x",
+                    parvars = c("a", "b"), family = "binomial",
+                    lx = -1, ux = 1, inipars = c(0, 7),
+                    iter = 100, k = 2,
+                    crtfunc = c_opt, sensfunc = c_sens,
+                    ICA.control = list(rseed = 1, checkfreq = Inf))
R> twoPL1

**************************************************************************
ICA iter: 100
 Points1  Points2
 -0.34277 0.34277
 Weights1 Weights2
  0.093    0.907
Criterion value:  0.4028266
Convergence: Maximum_Iteration
```



(a) `Output from plot(twoPL1)` (b) `Output from plot(twoPL2)`

Figure 4: Plots of the sensitivity functions of the generated $c-$optimal designs for estimating the ED95 when $x \in \chi = [-1, 1]$. The left panel (a) displays the sensitivity function of the locally $c-$optimal design when $\boldsymbol{\theta} = (0, 7)$. The right panel (b) displays the sensitivity function of the Bayesian $c-$optimal design when $a \sim U(-0.3, 0.3)$ and $b \sim U(6, 8)$.

```
CPU time: 5.173  seconds!
************************************************************************
************************************************************************
Maximum of the sensitivity function is  2.514716e-08
Efficiency lower bound (ELB) is  1
Verification required 0.504 seconds!
************************************************************************
```

The obtained design suggests that nearly 90% of the observations should be assigned to 0.34277 and the rest should be allocated to -0.34277. Figure 4 (a) displays the plot of the sensitivity function of the obtained design and it indicates its optimality. Using the given `c_opt()` and `c_sens()` functions, we can similarly find minimax $c-$optimal or robust designs. For illustrating example, see `?minimax` and `?robust`.

Finding Bayesian $c-$optimal design is very similar, except that each of (18) and (19) must be a vectorized R `function` with respect to the model parameters `a` and `b`:

```
R> c_opt_vec <-function(x, w, a, b, fimfunc){
+   gam <- log(.95/(1-.95))
+   M <- fimfunc(x = x, w = w, a = a, b = b)
+   B <- sapply(1:length(M), FUN = function(i)
+     matrix(c(1, -gam * b[i]^(-2)), ncol= 1) %*%
+       matrix(c(1, -gam * b[i]^(-2)), nrow = 1), simplify = FALSE)
+   sapply(1:length(M), FUN = function(i)
```

```
+     sum(diag(B[[i]] %*% solve(M[[i]])))
+ }
R> c_sens_vec <- function(xi_x, x, w, a, b, fimfunc){
+   gam <- log(.95/(1-.95)) # LD .95
+   M <- fimfunc(x = x, w = w, a = a, b = b)
+   M_inv <- lapply(M , FUN = function(FIM) solve(FIM))
+   M_x <- fimfunc(x = xi_x, w = 1, a = a, b = b)
+   B <- sapply(1:length(M), FUN = function(i)
+     matrix(c(1, -gam * b[i]^(-2)), ncol= 1) %*%
+       matrix(c(1, -gam * b[i]^(-2)), nrow = 1), simplify = FALSE)
+   sapply(1:length(M), FUN = function(i)
+     sum(diag(B[[i]] %*% M_inv[[i]] %*% M_x[[i]] %*% M_inv[[i]])) -
+       sum(diag(B[[i]] %*% M_inv[[i]])))
+ }
```

In the `c_opt_vec` and `c_sens_vec` functions, the arguments `a` and `b` are now vectors of the same (dynamic) length, and `fimfunc()` now returns a `list` of matrices with length equal to `length(a)`. Let $a \sim U(-0.3, 0.3)$ and $b \sim U(6, 8)$. Given `c_opt_vec` and `c_sens_vec`, the Bayesian $c-$optimal design for estimating the ED95 is obtained by

```
R> twoPL2 <- bayes(formula = ~1/(1 + exp(-b * (x-a))), predvars = "x",
+                  parvars = c("a", "b"), family = "binomial",
+                  lx = -1, ux = 1,
+                  prior = uniform(lower = c(-.3, 6), upper  = c(.3, 8)),
+                  iter = 100, k = 3,
+                  crtfunc = c_opt_vec,
+                  sensfunc = c_sens_vec,
+                  ICA.control = list(rseed = 1, ncount = 60, checkfreq = Inf),
+                  sens.bayes.control = list(cubature = list(maxEval = 100)))
R> print(twoPL2)

**************************************************************************
ICA iter: 100
 Points1  Points2  Points3
 -0.37216 0.02012  0.42577
 Weights1 Weights2 Weights3
  0.026    0.219    0.755
Criterion value:  0.6252608
Convergence: maxiter
CPU time: 117.193  seconds!
**************************************************************************
**************************************************************************
Maximum of the sensitivity function is  0.000523737
Efficiency lower bound (ELB) is  0.9997382
Verification required 2.12 seconds!
Adjust the control parameters in 'sens.bayes.control' for higher speed
**************************************************************************
```

Figure 4 (b) displays the plot of the sensitivity function of the design provided by the output and it verifies its optimality. Similar to the locally $c-$optimal design, this design puts more than 97% of its weight on the positive support points.

7. Summary

ICAOD modifies a state-of-the-art metaheuristic algorithm called Imperialist Competitive Algorithm to find different types of optimal designs for nonlinear models. We believe this package is more self-contained and has more capability than the few available in the literature. In particular, **ICAOD** offers different design approaches for handling the parameter dependency in the information matrix when the model is nonlinear. A useful feature of the **ICAOD** package is that it can create the Fisher information matrices for a very general class of nonlinear models automatically and also includes useful theory-based tools to assess proximity of any design to the optimal design without knowing the latter. Using **ICAOD**, it is also possible to find optimal designs for a user-specified optimality criterion, including hard-to-find various types of minimax optimal designs for which the criterion is not differentiable.

Due to space consideration, we presented only a few examples in this paper to show the functionality of the package. For additional applications, see Masoudi *et al.* (2017) and Masoudi *et al.* (2019) . The help-documentation manual for the package contains further details and illustrations. We hope that the generality and simplicity of the **ICAOD** package will encourage researchers from different disciplines to explore optimal design ideas in their work and enable them to implement a more informed design to realize maximum statistical efficiency at minimal cost.

Computational details

The results in this paper were obtained using R 3.5.3 with the **ICAOD** 0.9.9 package. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/`.

Acknowledgments

We would like to thank Dr. Paul-Christian Bürkner for his helpful comments when writing the package and this manuscript. The research of Wong reported in this paper was partially supported by a grant award R01GM107639 from the National Institute of General Medical Sciences of the National Institutes of Health. The research of Holling and Masoudi in this paper was partially supported by a grant (HO 1286/6 - 4) of the German Research Foundation (DFG). The contents in this paper are solely the responsibility of the authors and do not necessarily represent the official views of the National Institutes of Health.

References

Abdelbasit KM, Plackett RL (1983). "Experimental Design for Binary Data." *Jour-*

nal of the American Statistical Association, **78**(381), 90–98. ISSN 01621459. doi:10.1080/01621459.1983.10477936.

Atashpaz-Gargari E, Lucas C (2007). "Imperialist Competitive Algorithm: An algorithm for Optimization Inspired by Imperialistic Competition." In *IEEE congress on evolutionary computation*, pp. 4661–4667. doi:10.1109/CEC.2007.4425083.

Atkinson AC (1996). "The Usefulness of Optimum Experimental Designs." *Journal of the Royal Statistical Society B*, **58**, 59–76. doi:10.1111/j.2517-6161.1996.tb02067.x.

Atwood CL (1969). "Optimal and Efficient Designs of Experiments." *The Annals of Mathematical Statistics*, **40**(5), 1570–1602. doi:10.1214/aoms/1177697374.

Berger MP, King CJ, Wong WK (2000). "Minimax D-optimal Designs for Item Response Theory Models." *Psychometrika*, **65**(3), 377–390. doi:10.1007/BF02296152.

Berger MP, Wong WK (2005). *Applied optimal designs*. Jonh Wiley & Sons, Chichester, West Sussex, UK. doi:10.1002/0470857005.

Berger MP, Wong WK (2009). *An introduction to optimal designs for social and biomedical research*. Jonh Wiley & Sons, Chichester, West Sussex, UK. doi:10.1002/9780470746912.

Berntsen J, Espelid TO, Genz A (1991). "An Adaptive Algorithm for the Approximate Calculation of Multiple Integrals." *ACM Transactions on Mathematical Software (TOMS)*, **17**(4), 437–451. doi:10.1145/210232.210233.

Bürkner PC, Schwabe R, Holling H (2019). "Optimal designs for the generalized partial credit model." *British Journal of Mathematical and Statistical Psychology*. doi:10.1111/bmsp.12148.

Chaloner K, Larntz K (1989). "Optimal Bayesian Design Applied to Logistic Regression Experiments." *Journal of Statistical Planning and Inference*, **21**(2), 191–208. doi:10.1016/0378-3758(89)90004-9.

Chaloner K, Verdinelli I (1995). "Bayesian Experimental Design: A Review." *Statistical Science*, **10**(3), 273–304. doi:10.1214/ss/1177009939.

Chernoff H (1953). "Locally Optimal Designs for Estimating Parameters." *The Annals of Mathematical Statistics*, **24**(4), 586–602. doi:10.1214/aoms/1177728915.

Clyde M, Müller P, Parmigiani G (1995). "Optimal design for heart defibrillators." In *Case Studies in Bayesian Statistics, Volume II*, pp. 278–292. Springer. doi:10.1007/978-1-4612-2546-1_7.

Dette H, Kiss C, Bevanda M, Bretz F (2010). "Optimal designs for the EMAX, log-linear and exponential models." *Biometrika*, **97**(2), 513–518. doi:10.1093/biomet/asq020.

Dragalin V, Hsuan F, Padmanabhan SK (2007). "Adaptive designs for dose-finding studies based on sigmoid e max model." *Journal of Biopharmaceutical Statistics*, **17**(6), 1051–1070. doi:10.1080/10543400701643954.

Fedorov V (1980). "Convex Design Theory." *Series Statistics*, **11**(3), 403–413. doi:10.1080/02331888008801549.

Fedorov VV, Hackl P (2012). *Model-Oriented Design of Experiments*, volume 125. Springer Science & Business Media. doi:10.1007/978-1-4612-0703-0.

Fedorov VV, Leonov SL (2013). *Optimal design for nonlinear response models*. CRC Press.

Firth D, Hinde J (1997). "On Bayesian D-optimum Design Criteria and the Equivalence Theorem in Non-linear Models." *Journal of the Royal Statistical Society B*, **59**(4), 793–797. doi:10.1111/1467-9868.00096.

Gablonsky JM, Kelley CT (2001). "A Locally-biased Form of The DIRECT Algorithm." *Journal of Global Optimization*, **21**(1), 27–37. doi:10.1023/A:1017930332101.

Genz A, Malik A (1980). "Remarks on Algorithm 006: An Adaptive Algorithm for Numerical Integration over an N-dimensional Rectangular Region." *Journal of Computational and Applied Mathematics*, **6**(4), 295 – 302. ISSN 0377-0427. doi:10.1016/0771-050X(80)90039-X.

Graßhoff U, Holling H, Schwabe R (2012). "Optimal designs for the Rasch model." *Psychometrika*, **77**(4), 710–723. doi:10.1007/s11336-012-9276-2.

Holling H, Schwabe R (2013). "An introduction to optimal design: Some basic issues using examples from dyscalculia research." *Zeitschrift für Psychologie*, **221**(3), 124.

Hosseini S, Al Khaled A (2014). "A Survey on the Imperialist Competitive Algorithm Metaheuristic: Implementation in Engineering Domain and Directions for Future Research." *Applied Soft Computing*, **24**, 1078–1094. doi:10.1016/j.asoc.2014.08.024.

Hyun SW, Wong WK, Yang Y (2018). "VNM: An R Package for Finding Multiple-Objective Optimal Designs for the 4-Parameter Logistic Model." *Journal of Statistical Software*, **83**(5), 1–19. doi:10.18637/jss.v083.i05.

Johnson SG (2013). "Cubature Package." Version 1.0.3, URL https://github.com/stevengj/cubature.

Johnson SG (2014). "The NLopt Nonlinear-Optimization Package." URL http://ab-initio.mit.edu/nlopt.

Kennedy J, Eberhart R (1995). "Particle swarm optimization." In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pp. 1942–1948 vol.4. doi:10.1109/ICNN.1995.488968.

Kiefer J, Wolfowitz J (1959). "Optimum Designs in Regression Problems." *The Annals of Mathematical Statistics*, **30**, 271–294. doi:10.1214/aoms/1177706252.

King J, Wong WK (2000). "Minimax D-Optimal Designs for the Logistic Model." *Biometrics*, **56**(4), 1263–1267.

Lehmann EL, Casella G (1998). *Theory of point estimation*, volume 31. Springer Science & Business Media.

Lin JL, Cho CW, Chuan HC (2013). "Imperialist Competitive Algorithms with Perturbed Moves for Global Optimization." In *Applied Mechanics and Materials*, volume 284, pp. 3135–3139. Trans Tech Publ. doi:10.4028/www.scientific.net/amm.284-287.3135.

Macdougall J (2006). "Analysis of Dose–Response Studies–E max Model." In *Dose finding in drug development*, pp. 127–145. Springer, New York, NY. doi:10.1007/0-387-33706-7_9.

Masoudi E, Holling H, Duarte BP, Wong WK (2019). "Metaheuristic Adaptive Cubature Based Algorithm to Find Bayesian Optimal Designs for Nonlinear Models." *Journal of Computational and Graphical Statistics*. doi:10.1080/10618600.2019.1601097.

Masoudi E, Holling H, Wong WK (2016). "ICAOD: Optimal Designs for Nonlinear Models." R package version 0.9.8, URL http://CRAN.R-project.org/package=ICAOD.

Masoudi E, Holling H, Wong WK (2017). "Application of Imperialist Competitive Algorithm to Find Minimax and Standardized Maximin Optimal Designs." *Computational Statistics & Data Analysis*, **113**, 330–345. doi:10.1016/j.csda.2016.06.014.

Masoudi E, Sarmad M, Talebi H (2013). *LDOD: Finding Locally D-optimal Optimal Designs for Some Nonlinear and Generalized Linear Models.* R package version 1.0, URL http://CRAN.R-project.org/package=LDOD.

Narasimhan B, Johnson SG (2017). "cubature: Adaptive Multivariate Integration over Hypercubes." R package version 1.3-11, URL https://CRAN.R-project.org/package=cubature.

Overstall A, Woods D, Adamou M (2017). "acebayes: An R Package for Bayesian Optimal Design of Experiments via Approximate Coordinate Exchange." URL arXivpreprintarXiv:1705.08096.

Overstall AM, Woods DC (2017). "Bayesian Design of Experiments Using Approximate Coordinate Exchange." *Technometrics*, **59**(4), 458–470. doi:10.1080/00401706.2016.1251495.

Pukelsheim F, Rieder S (1992). "Efficient Rounding of Approximate Designs." *Biometrika*, **79**(4), 763–770. doi:10.2307/2337232.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

SAS Institute Inc (2016). *JMP® 13 Design of Experiments Guide*. Cary, NC: SAS Institute Inc. URL https://www.jmp.com.

Silvey SD (1980). *Optimal design : an introduction to the theory for parameter estimation*. London; New York : Chapman and Hall. doi:10.1007/978-94-009-5912-5.

Sitter RR (1992). "Robust Designs for Binary Data." *Biometrics*, **48**(4), 1145–1155. doi:10.2307/2532705.

Thomas N (2006). "Hypothesis Testing and Bayesian Estimation using a Sigmoid E max Model Applied to Sparse Dose-Response Designs." *Journal of Biopharmaceutical Statistics*, **16**(5), 657–677. doi:10.1080/10543400600860469.

Wang T, Yang M (2014). "Adaptive optimal designs for dose-finding studies based on sigmoid Emax models." *Journal of Statistical Planning and Inference*, **144**, 188–197. doi:10.1016/j.jspi.2013.09.003.

Weiser C (2016). "mvQuad: Methods for Multivariate Quadrature." R package version 1.0-6, URL http://CRAN.R-project.org/package=mvQuad.

Wong WK (1992). "A Unified Approach to the Construction of Minimax Designs." *Biometrika*, **79**(3), 611–619. doi:doi.org/10.1093/biomet/79.3.611.

Yang XS (2011). "Metaheuristic Optimization: Algorithm Analysis and Open Problems." In PM Pardalos, S Rebennack (eds.), *Experimental Algorithms*, pp. 21–32. Springer-Verlag, Berlin. doi:10.1007/978-3-642-20662-7_2.

Zhu W, Wong WK (2001). "Bayesian optimal designs for estimating a set of symmetrical quantiles." *Statistics in Medicine*, **20**(1), 123–137. doi:10.1002/1097-0258(20010115)20:1<123::aid-sim643>3.0.co;2-5.

Affiliation:

Ehsan Masoudi
Department of Psychology, University of Münster
Fliednerstr. 21, 48149 Germany
E-mail: ehsan.masoudi@uni-muenster.de
and
Heinz Holling
Department of Psychology, University of Münster
Fliednerstr. 21, 48149 Germany
E-mail: holling@uni-muenster.de
and
Weng Kee Wong
Department of Biostatistics
UCLA Fielding School of Public Health
Los Angeles, CA 90095-1772, USA
E-mail: wkwong@ucla.edu