

dfpk An R-package for Bayesian dose-finding designs using pharmacokinetics (PK) for phase I clinical trials

A. Toumazi, E. Comets, C. Alberti, T. Friede, F. Lentz, N. Stallard, S. Zohar, M. Ursino

▶ **To cite this version:**

A. Toumazi, E. Comets, C. Alberti, T. Friede, F. Lentz, et al.. dfpk An R-package for Bayesian dose-finding designs using pharmacokinetics (PK) for phase I clinical trials. Computer Methods and Programs in Biomedicine, 2018, 157, pp.163-177. 10.1016/j.cmpb.2018.01.023 . hal-01737106

HAL Id: hal-01737106
https://univ-rennes.hal.science/hal-01737106

Submitted on 18 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Computer Methods and Programs in Biomedicine 157 (2018) 163–177

Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

journal homepage: www.elsevier.com/locate/cmpb

dfpk: An R-package for Bayesian dose-finding designs using pharmacokinetics (PK) for phase I clinical trials

A. Toumazi[a], E. Comets[b,c], C. Alberti[d], T. Friede[e], F. Lentz[f], N. Stallard[g], S. Zohar[a,1], M. Ursino[a,1,*]

[a] INSERM, UMRS 1138, Team 22, CRC, University Paris 5, University Paris 6, Paris, France
[b] INSERM, CIC 1414, University Rennes-1, Rennes, France
[c] INSERM, IAME UMR 1137, University Paris Diderot, Paris, France
[d] INSERM, UMR 1123, Hôpital Robert-Debré, APHP, University Paris 7, Paris, France
[e] Department of Medical Statistics, University Medical Center Göttingen, Göttingen, Germany
[f] Federal Institute for Drugs and Medical Devices, Bonn, Germany
[g] Statistics and Epidemiology, Division of Health Sciences, Warwick Medical School, The University of Warwick, UK

ARTICLE INFO

Article history:
Received 30 June 2017
Revised 11 January 2018
Accepted 24 January 2018

Keywords:
Dose-finding
Maximum tolerated dose
Pharmacokinetics
Phase I clinical trials
R package

ABSTRACT

Background and objective: Dose-finding, aiming at finding the maximum tolerated dose, and pharmacokinetics studies are the first in human studies in the development process of a new pharmacological treatment. In the literature, to date only few attempts have been made to combine pharmacokinetics and dose-finding and to our knowledge no software implementation is generally available. In previous papers, we proposed several Bayesian adaptive pharmacokinetics-based dose-finding designs in small populations. The objective of this work is to implement these dose-finding methods in an R package, called `dfpk`.

Methods: All methods were developed in a sequential Bayesian setting and Bayesian parameter estimation is carried out using the `rstan` package. All available pharmacokinetics and toxicity data are used to suggest the dose of the next cohort with a constraint regarding the probability of toxicity. Stopping rules are also considered for each method. The `ggplot2` package is used to create summary plots of toxicities or concentration curves.

Results: For all implemented methods, `dfpk` provides a function (`nextDose`) to estimate the probability of efficacy and to suggest the dose to give to the next cohort, and a function to run trial simulations to design a trial (`nsim`). The `sim.data` function generates at each dose the toxicity value related to a pharmacokinetic measure of exposure, the AUC, with an underlying pharmacokinetic one compartmental model with linear absorption. It is included as an example since similar data-frames can be generated directly by the user and passed to `nsim`.

Conclusion: The developed user-friendly R package `dfpk`, available on the CRAN repository, supports the design of innovative dose-finding studies using PK information.

© 2018 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license.
(http://creativecommons.org/licenses/by-nc-nd/4.0/)

1. Introduction

Dose-finding studies and pharmacokinetics (PK) are carried out at the first phases of clinical evaluation of a new drug in humans. Drug safety is evaluated in the dose-finding study, which aims at identifying the maximum tolerated dose (MTD) [1]. Meanwhile, the PK data collected during such study provides the description of the dose-concentration relationships [2]. Nevertheless, these two approaches are often conducted and reported independently in different sections in publications reporting trial results [3]. Identifying the right dose or set of doses at an early stage is crucial: selecting too toxic doses can result in patient overdosing, while selecting an inefficient dose increases the likelihood that the drug will be found to be ineffective in subsequent clinical evaluation [4]. Particularly in the case of small populations, such as rare diseases or paediatrics, it should be useful to take into account all the infor-

* Corresponding author..
E-mail address: moreno.ursino@inserm.fr (M. Ursino).
[1] The last two authors contributed equally as co-senior authors.

https://doi.org/10.1016/j.cmpb.2018.01.023
0169-2607/© 2018 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license. (http://creativecommons.org/licenses/by-nc-nd/4.0/)

mation collected during the trial, and to try to utilize the PK measurements within the dose-finding design. Only few attempts have been described in the literature so far and, usually, the methods were built for a very specific situation [5–8]. Moreover, no software implementations are publicly available.

In this article we present the new R package `dfpk` (short for dose-finding Pharmacokinetics), which provides the Bayesian adaptive PK-based dose-finding designs in small populations proposed by Ursino et al. [8] through the freely available R software [9]. The six methods detailed in [8] have been implemented in `dfpk`. For each of them, two functions are provided: (i) a function to determine the next recommended dose (during the trial) or the recommended MTD (at the end of the trial) and (ii) a function to run simulations of phase I studies to design a new trial. Interactive graphical representations of the dose-concentration curve, of the dose allocation process in the trial and of the dose-toxicity response are also provided by the package.

The paper is organised as follows. Section 2 introduces the statistical methods proposed by Ursino et al. [8], along with the description of the suggested scenarios to be simulated. Section 3 outlines the structure of the package and the main functions of the paper (`sim.data`, `nextDose` and `nsim`) with practical examples. Section 4 & 5 include conclusion, discussion and recommendations.

2. Computational methods

The present section briefly reviews the methods proposed by Ursino et al. [8] to perform dose-finding taking into account PK measurements. We then describe the scenarios simulated in [8] in order to evaluate the robustness of the method, which have been added as examples in the dfpk package.

2.1. Dose-finding methods

Let $D = \{d_1, \ldots, d_k\}$ be the set of K possible doses with $d_1 < \cdots < d_k$ and $d_{[i]} \in D$ be the dose administered to the ith subject ($i = 1, \ldots, n$, where n denotes the sample size) and y_i be a binary variable which takes value 1 if the ith subject shows a DLT (dose-limiting toxicity) and 0 otherwise. Moreover, let z_i be the logarithm of the area under the curve (AUC) of the concentrations of drug in blood plasma against time, for the ith patient.

All methods share the same fundamental idea for the dose-escalation rule: the dose chosen for the next cohort enrolled is the one with probability of toxicity nearest to the target θ selected by the trial investigators. A no-skipping rule is given: if some doses have not yet been tested, the dose is chosen from $D^* \subset D$, a subset of D which contains all the doses already evaluated and the first dose level immediately above. The final recommended MTD is given by the dose that would have been administered for the $(n+1)$st subject enrolled in the trial. Finally, we added in all methods the same stopping rule: if the posterior probability of toxicity of the first dose is greater of a specified threshold, then no dose is suggested and the trial is stopped.

Each method is separated from the others. We adopted the convention of starting the subscription of β parameter from 0 for each method. Therefore, even if the parameters share the same names across models, they have different interpretations. In the following, we briefly describe how the probability of toxicity is estimated and computed in each method.

2.1.1. PKCOV

PKCOV is a modification of the method proposed by Piantadosi and Liu [5] who suggested to use the AUC as a covariate for p_T, the probability of toxicity, through the logit link. Therefore, the dose-toxicity model is

$$\text{logit}(p_T(d_k, \Delta z_{dk}, \beta)) = -\beta_0 + \beta_1 \log(d_k) + \beta_2 \Delta z_{dk} \quad \forall d_k \in D, \tag{1}$$

where $\boldsymbol{\beta} = (\beta_1, \beta_2)$, β_0 is a constant selected through a sensitivity analysis or with prior information, Δz_{dk} is the difference between the logarithm of population AUC at dose d_k and z, the logarithm of AUC of the subject at the same dose. Independent uniform distributions are selected as prior distributions for β_1 and β_2. In detail, $\beta_1 \sim U(\max(0, m_1 - 5), m_1 + 5)$, where m_1 reflects the prior information on the parameter and the length of the domain of the distribution can go up to 10, and $\beta_2 \sim U(0, 5)$. Both β_0 and m_1 should be selected using prior information, such as from preclinical data, and sensitivity analysis should be done. The estimated probability of toxicity versus dose is obtained by inverting Eq. (1), using $\beta_1 = \hat{\beta}_1$, the estimated parameter, and $\Delta z_{dk} = 0$.

2.1.2. PKLIM and PKCRM

PKLIM is a modification of the method proposed by Patterson et al. [6] and Whitehead et al. [10]. A normal PK-toxicity model is used:

$$z_i | \boldsymbol{\beta}, \nu \sim N\left(\beta_0 + \beta_1 \log d_i, \nu^2\right), \tag{2}$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1)$ are the regression parameters, and ν is the standard deviation. A bivariate normal distribution and a beta distribution are chosen for $\boldsymbol{\beta}$ and ν, respectively, that is, $\boldsymbol{\beta} \sim N(\mathbf{m}, \nu^2(g\mathbb{1}))$ and $\nu \sim Beta(1, 1)$. Therefore, a hierarchical prior distribution is given to $\boldsymbol{\beta}$, where $\mathbf{m} = (-\log Cl_{pop}, 1)$ and g should be chosen using prior information. For instance, Cl_{pop} denotes the attended value of the clearance at population level, and g reflects the precision. The probability of toxicity of each dose is computed as

$$P(z > L \mid d_k, \boldsymbol{\beta} = \hat{\boldsymbol{\beta}}, \nu = \hat{\nu}) \quad \forall d_k \in D, \tag{3}$$

where L is a threshold set before starting the trial and the hat denotes the posterior means of the parameters. Since an assumption underlying the model is that DLTs are based on the AUC exceeding some threshold, the method could be applicable only when such a threshold is known. In order to avoid this problem, the PKCRM method was proposed, which is the combination of PKLIM and the CRM [11] using a power working model and normal prior on the parameter. In PKCRM the dose recommended for the next subject is the lowest of the doses recommended by the two methods.

Note that although the same notation has been used for convenience, the parameters β_0 and β_1 are different in the different models.

2.1.3. PKLOGIT, PKPOP, PKTOX

PKLOGIT, inspired by Whitehead et al. [7], combines two regressions to compute the probability of toxicity versus the dose. The first one is the same as Eq. (2), that is z versus dose. In the second, z is used as a covariate in a logistic regression model for p_T. This means that now the probability of toxicity is described in term of AUC and not any more in term of dose. Therefore, we have that

$$\text{logit}(p_T(z, \boldsymbol{\beta})) = -\beta_2 + \beta_3 z, \tag{4}$$

where β_2 and β_3 have independent uniform prior distributions, that is, $\beta_2 \sim U(0, m_2)$ and $\beta_3 \sim U(0, m_3)$, with $m_2 \geq m_3$, and values can be chosen using prior information. If no information is available, $m_2 = 20$ and $m_3 = 10$ are good starting values for a sensitivity analysis. The probability of toxicity associated with each dose is obtained by using the estimated parameters of each regression model in the following expected value formula:

$$P(y = 1 \mid d_k, \boldsymbol{\beta} = \hat{\boldsymbol{\beta}}, \nu = \hat{\nu}) = E\left[\frac{1}{1 + e^{\hat{\beta}_2 - \hat{\beta}_3 z}}\right] = \int \frac{1}{1 + e^{\hat{\beta}_2 - \hat{\beta}_3 z}} g(z)\, dz, \tag{5}$$

where $g(z)$ represents the distribution of the logarithm of AUC given the dose d_k obtained from Eq. (2).

PKPOP, a variation of PKLOGIT, arises by replacing z with $z_{k,\,pop}$ in Eq. (4), where $z_{k,\,pop}$ is the mean value of the logarithm of AUC at dose d_k predicted by Eq. (2). In other words, we replace the observed AUC value for the patient with the population mean value. Then, the probability of toxicity at each dose is computed inverting Eq. (4), using the estimated parameters $\hat{\beta}_2$ and $\hat{\beta}_3$ along with $z_{k,\,pop}$ predicted by Eq. (2).

PKTOX is essentially the PKLOGIT method with a probit regression model replacing the logistic regression in Eq. (4), that is

$$p_T(z, \boldsymbol{\beta}) = \Phi(-\beta_2 + \beta_3 z), \tag{6}$$

with Φ represents the standard cumulative normal distribution. As in the previous models, independent uniform distributions are chosen as prior distributions for the parameters. The probability of toxicity versus dose is then computed in the same way of Eq. (5) using the probit regression inside the integral.

2.1.4. DTOX

DTOX follows the usual way of estimating p_T versus dose directly without the PK measurements and is included to check the behaviour of this standard method. The dose-toxicity model is:

$$p_T(d_k, \boldsymbol{\beta}) = \Phi(-\beta_0 + \beta_1 \log(d_k)) \quad \forall d_k \in D. \tag{7}$$

Independent uniform bivariate prior distribution is chosen for $\boldsymbol{\beta}$ in the same way of Eq. (4).

2.2. Simulating data for trial design

The package implements several examples reproducing the scenarios proposed in Ursino et al. [8] to evaluate the method performance using simulated data. In these settings, toxicity is linked to a PK measure of exposure, namely to AUC, and we used a simulation setting similar to the one in [12]. A first order absorption, linear, one compartment PK model was used to simulate PK data. In this model, the concentration at time t after administration of a dose d_k of the drug at time 0, can be written as a function of k_a, the absorption rate constant for oral administration, CL, the clearance of elimination, and V, the volume of distribution, as follows:

$$c(t) = \frac{d_k}{V} \frac{k_a}{k_a - CL/V} \left(e^{-(CL/V)t} - e^{-k_a t} \right), \tag{8}$$

Individual CL and V are sampled from log-normal distributions with mean $CL_{pop}(Lh^{-1})$ and $V_{pop}(L)$, respectively, and standard deviation $\omega_{CL} = \omega_V$, while k_a is fixed in this study as limited information concerning the absorption phase was available in the original dataset to estimate its inter-individual variability. In order to link PK to the toxicity profile of patients, a sensitivity parameter α, coming from a log-normal distribution with mean equal to 1 and standard deviation ω_α, and a threshold τ_T are introduced. We assumed that a patient incurs a dose limiting toxicity (DLT) if $\alpha AUC \geq \tau_T$. Choosing the different parameters leads to several scenarios. The probability of toxicity is computed as:

$$p_T(d_k) = \Phi\left(\frac{\log d_k - \log \tau_T - \log CL}{\sqrt{\omega_{CL}^2 + \omega_\alpha^2}} \right). \tag{9}$$

3. R-functions

Package `dfpk` implements all the dose-finding methods described in Section 2. In Fig. 1, an overall explanation of the main functions in the `dfpk` package is given. The aim of the package is to assist the design of phase I clinical trials. During the trial, the patient data already accrued ("Trial data" in Fig. 1A) can be used in the `nextDose` function in order to determine the next

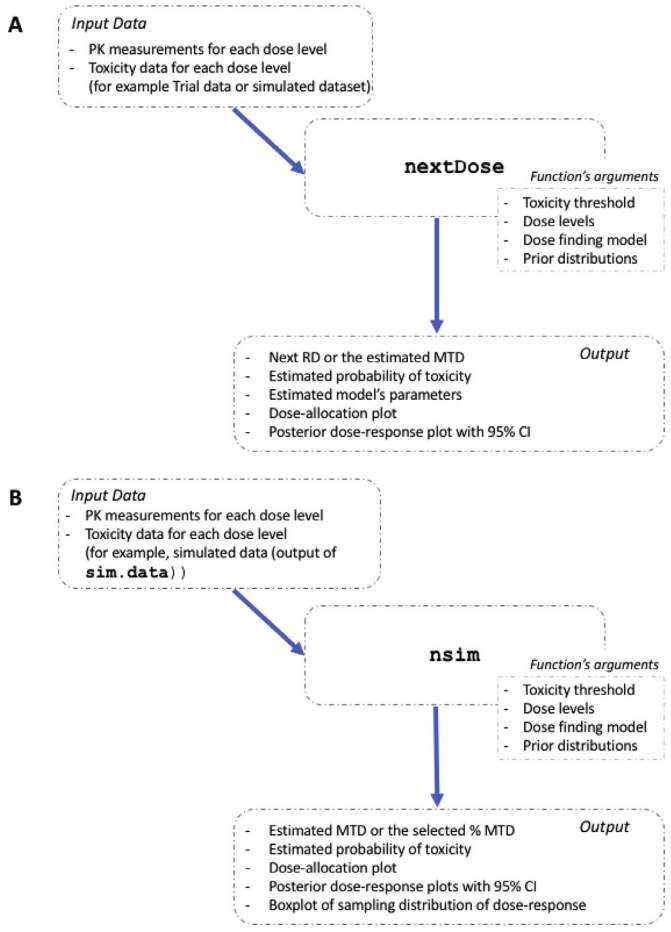


Fig. 1. Overall presentation of the main R functions `nextDose` (Fig. 1A) and `nsim` (Fig. 1B) available in the package dfpk, indicating the corresponding inputs and outputs.

recommended dose, or the MTD at the end of trial. Plots are also available after the estimation process. When planning a new trial, datasets containing PK and toxicity measurements can be simulated directly by the user ("Simulated data") or through the `sim.data` function, and used in the `nsim` function (in Fig. 1B.), which will perform n simulated clinical trials. Also in this case, plots with graphical representations support the numeric results.

Bayesian parameter estimation is carried out using the `rstan` package while the `ggplot2` package is used to create plots. Three S4 classes are implemented in the package, *"dose"* , *"dosefinding"* and *"scen"*, in order to store the outputs of the main R functions `nextDose`, `nsim` and `sim.data`, respectively. The classes are detailed described in the Appendix B.

The package `dfpk` is available on the CRAN archive and can be easily installed on the fly through the URL https://cran.r-project.org/web/packages/dfpk. Once the package is installed, it can be loaded with the command:

```
> library(dfpk)
```

In the remainder of this section, we present the R functions available in the package along with the required input parameters and examples. A more extensive demonstration and documentation can be accessed from the on-line user manual on the CRAN server by installing the package or directly within the R console.

Table 1
Arguments for the functions `nextDose` and the dose-finding models.

model	The dose-finding model chosen between "pktox", "pkcov", "pkcrm", "pklogit", "pkpop" and "dtox".
y	A binary vector of toxicity outcomes from previous patients.
AUCs	A vector with the computed AUC values of each previous patient for PKTOX, PKCRM, PKLOGIT and PKPOP.
doses	A vector with the doses panel.
x	A vector with the dose level assigned to previous patients.
theta	The toxicity target.
options	List with the Stan model's options.
prob	The threshold of the posterior probability of toxicity for the stopping rule in the selected model; defaults to 0.9.
betapriors	A vector with the value for the prior distribution of the regression parameters in the selected model.
thetaL	A second threshold of AUC in the PKCRM model only; defaults to theta for PKCRM and NULL for the models PKTOX, PKCOV, PKLOGIT, PKPOP & DTOX.
*p*0	The skeleton of CRM for PKCRM; defaults to NULL.
L	The AUC threshold to be set before starting the trial for PKCRM; defaults to NULL.
deltaAUC	A vector of the difference between computed individual previous patients' AUC and the AUC of population at the same dose level (defined as an average); argument for PKCOV; defaults to NULL.
CI	A logical constant indicating the estimation of the 95% credible intervals (CI) of the probability of toxicity at each dose level for the selected model; defaults to TRUE.

Table 2
Input arguments required by each dose-finding method in the `nextDose` function.

Method	Required arguments	Optional arguments
pktox	y, AUCs, doses, x, theta, prob, options, CI	betapriors
pkcov	y, AUCs, doses, x, theta, deltaAUC, prob, options, CI	betapriors
pkcrm	y, AUCs, doses, x, theta, p0, L, prob, thetaL, options, CI	betapriors
pklogit	y, AUCs, doses, x, theta, prob, options, CI	betapriors
pkpop	y, AUCs, doses, x, theta, prob, options, CI	betapriors
dtox	y, doses, x, theta, prob, options, CI	betapriors

3.1. Dose-finding methods (*nextDose*)

The `nextDose` function is used to perform parameter estimation at each step during a dose-finding trial. It gives the recommended dose to administer to the next cohort of patients, or the final estimated MTD if applied at the end of the trial. It can be used during an ongoing clinical trial or with a simulated dataset, as described later in the Section 3.2.

```
> nextDose(model, y, AUCs, doses, x, theta, options, prob = 0.9, betapriors = NULL,
            thetaL = NULL, p0 = NULL, L = NULL, deltaAUC = NULL, CI = TRUE)
```

The description of the input arguments used in the `nextDose` function are provided in Table 1. The user has to choose the dose-finding method from the available set in the `model` parameter. Then, he/she should provide the parameters required by the selected method, as specified in Table 2, while the others are set automatically to NULL. Any argument not specified by the user will be set to the corresponding default choice [8]. The number of chains and iterations for the Bayesian algorithm can be changed using the appropriate `rstan` options.

3.1.1. Demonstration

(A) PKTOX model In the following example, we supposed that a clinical trial is still ongoing and 15 patients have been enrolled so far. For this case, PKTOX was selected as the dose-finding model. The `nextDose` function requires the following input arguments: the panel of doses of the drug, the target toxicity probability, θ and the options for the Stan model as a list, containing the number of chains, the number of iterations and the warm-up iterations.

```
> doses <- c(12.59972,34.65492,44.69007,60.80685,83.68946,100.37111)
> theta <- 0.2
> options <- list(nchains = 4, niter = 4000, nadapt = 0.9)
```

Other necessary input arguments are the vector of dose levels assigned to the patients (as integer vector), that is denoted by x, the AUCs values and the vector of toxicity outcomes (0/1 are accepted), denoted as y, for each enrolled patient.

```
> AUCs <- c(1.208339,  5.506040,  6.879835,  3.307928,  3.642430,
            10.271291,  3.885522,  3.086622,  2.537158,  5.525917,
            8.522176,  4.642741, 11.048531, 10.246976,  5.226807)
> x <- c(1, 2, 3, 4, 5, 6, 4, 4, 4, 5, 5, 4, 4, 5, 5)
> y <- c(0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0)
```

The user can change the prior distribution parameter of the toxicity-AUC regression by adding the parameter betapriors. If it is not specified, the value suggested by Ursino et al. [8] are used by default.
The default choices of betapriors for PKTOX model are the following:

$$\begin{aligned} \beta|\nu &\sim N(m, \nu \times g \times diag(1,1)),\\ \nu &\sim Beta(1,1),\\ m &= \left(-\log\left(Cl_{pop}\right), 1\right),\\ \beta_2 &\sim U(0, beta2mean),\\ \beta_3 &\sim U(0, beta3mean), \end{aligned} \tag{10}$$

where Cl_{pop} is the population clearance and the default choices are $Cl_{pop} = 10$, $g = 10000$, beta2mean = 20 and beta3mean = 10. Details about the default choices of all the dose-finding models are available in the user-manual on the CRAN.

These arguments are used in the nextDose function depending on the chosen model, PKTOX, using the following syntax:

```
> nextD <- nextDose(model = "pktox", y=y, AUCs=AUCs, doses=doses,
                    x=x, theta=theta, options=options)
```

omitting all default parameters.
The results are stored in a *"dose"* object. The output below reports part of the results displayed, that is the number of patients who are currently enrolled, the selected dose-finding model, and the observed dose levels of the drug:

```
Results of an ongoing clinical trial

Model: pktox

Total number of enrolled patients in the trial:  15

Levels of doses:  12.59972 34.65492 44.69007 60.80685 83.68946 100.3711
```

According to these results, the next recommended dose level is 5, which would be the dose for the next cohort of patients given the data. The estimated toxicity probabilities and model parameters are also shown:

```
The Next Recommended Dose:

[1] 5

Estimated probability of toxicity:

[1] 0.0004 0.0225 0.0473 0.1026 0.1984 0.2714


Estimated model's parameters:

     beta0      beta1         nu      beta2      beta3
-1.5590477  0.7709732  0.5266802 -9.0637545  3.8542888
```

The package also provides a graphical representation of the results. For example, using the generic function plot() on a *"dose"* object, we can select if we want to present graphically: (1) the dose allocation of the currently enrolled patients during the trial or (2) the posterior distributions of the probability of toxicity at each dose presenting the estimation along with the lines of the 95% credible intervals (CI), as below:

```
> plot(nextD)
```

Fig. 2 presents (A) the data for the first 15 patients in the study, and (B) the plot of the posterior distributions given this data of the probability of toxicity at each dose according to the PKTOX method (including the mean estimation along with the 95% CI).

(B) PKCRM model

In this section a second illustration is shown using the PKCRM dose-finding method in the function nextDose. In this way, we can highlight the differences in the required inputs/outputs. In this case, in addition to the input arguments that are used in the PKTOX

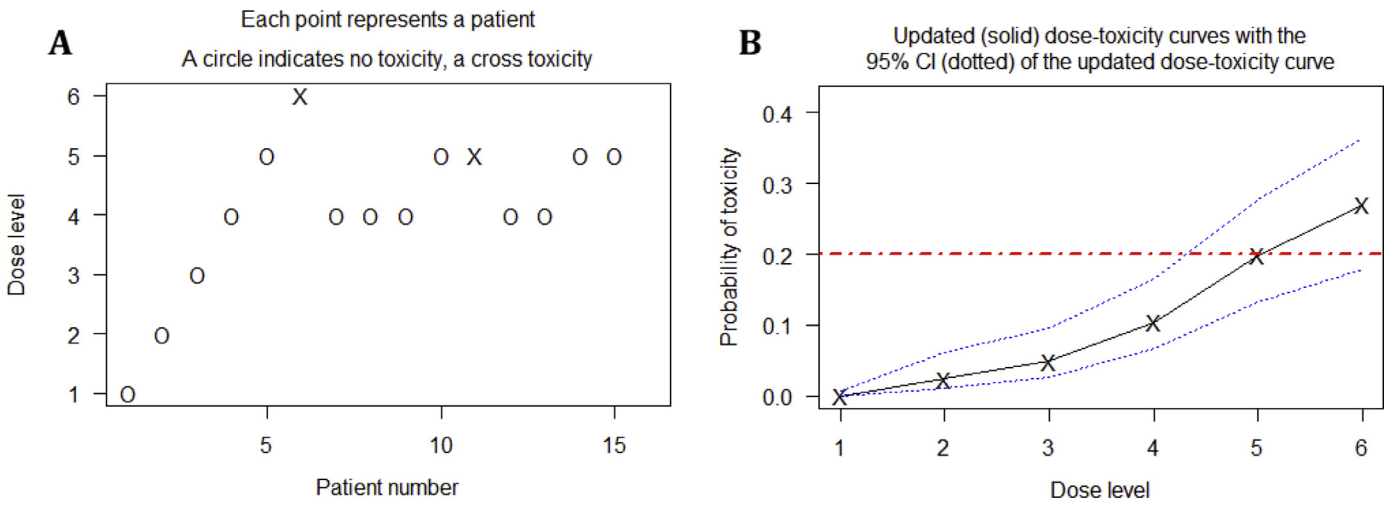


Fig. 2. (A) Plot of the data for the first 15 patients in the study and, (B) the plot of the posterior distributions given this data of the probability of toxicity at each dose according to the PKTOX method (including the mean estimation along with the 95% CI).

method (i.e. y, AUCs, doses, x, theta, options), the PKCRM method requires the skeleton of CRM (p0) and the AUC threshold (L), which must be set before starting the trial. In this example we use:

```
> p0 <- c(.01,.05,.1,.2,.35,0.45)
> L <- log(15.09)
```

After setting all the required arguments for the chosen PKCRM model, we call the nextDose function as following:

```
> nextDose(model = "pkcrm", y=y, AUCs=AUCs,
                    doses=doses, x=x, theta=theta,
            options=options, p0 = p0, L = L)
```

The resulting object displays as follows:

```
Results of an ongoing clinical trial
Model: pkcrm
Total number of enrolled patients in the trial:  15
Levels of doses:  12.59972 34.65492 44.69007 60.80685 83.68946 100.3711

The Next Recommended Dose:
[1] 5


Estimated probability of toxicity:
[1] 0.0008 0.0178 0.0329 0.0640 0.1166 0.1576

Estimated model's parameters:
     beta0      beta1         nu
-1.4750896  0.7504786  0.5286834
```

3.2. *Generate data (sim.data)*

The `sim.data` function generates and stores PK and toxicity data in order to be used for simulation according to the model described in Section 2.2. The initial step consists in generating the patients' responses at all doses for each trial. The function `sim.data` takes the trial's settings and returns a list of data including the subject's PK parameters along with the simulated toxicity observations.

```
> sim.data(PKparameters, omegaIIV, omegaAlpha,
           sigma, doses, limitTox,
           timeSampling, N, TR, seed=190591)
```

In particular, it starts by drawing subject's PK parameters (k_α, CL and V) from the population distributions defined by the population mean (`PKparameters`) and the inter-individual variability (IIV) for the clearance and the volume of distribution (`omegaIIV`). Then, for each dose level (`doses`), we computed the desired probability of toxicity (`preal`), and for all patients (`N`) and simulated data (`TR`), it computes the concentration measurements at the specified time points (`timeSampling`). In addition, we add a proportional error drawn from a normal distribution with zero mean and a standard deviation `sigma`. According to eq. (9), the function computes the toxicity values for each dose level using the threshold `limitTox` and the patient's sensitivity parameter `omegaAlpha`. Finally, a default value of the random number generator (`seed`) is set at 190591.

The results are stored in a list of *"scen"* objects, which consists of `PKparameters`, `nPK`, the length of the time points, `timeSampling`, `idtr`, `N`, `doses`, `preal`, `limitTox`,`omegaIIV`, `omegaAlpha`, `concentration`, the concentration computed at the PK population values, `concPred`, the concentration values with proportional errors for each patient at each dose, `tox`, `tab`, a summary matrix, used in the simulation function, containing the sampling time points at the first row followed by `concPred`, `parameters`, the simulated PK parameters of each patient, `alphaAUC`, the αAUCs. Since we are using S4 classes, the user can easily create his/her own datasets. For example, he/she can create a new object for each simulated trial, named `UserData`, using the command `new` in the following way

```
>  UserData <- new("scen", PKparameters, nPK,
                   timeSampling, idtr, N, doses, preal,
                   limitTox, omegaIIV, omegaAlpha,
                   concentration, concPred, tox, tab,
                   parameters, alphaAUC)
```

and then add it in a list, along with the other simulated datasets. A more expanded description of the *"scen"* class and objects can be accessed from Appendix B2.

3.2.1. Demonstration

The following illustration shows how to generate the datasets of the first scenario described by Ursino et al. [8]. In this example, we set the number of trials to 10 (i.e. TR=10), the threshold of toxicity value to 10.96, the dose levels to (12.6, 34.655, 44.69, 60.807, 83.689 and 100.371 mg) which are used to obtain the true probabilities of toxicity, 48 evenly spaced sampling time points between 0 and 48 h, a sample size of 30 and $k_a = 2$, $CL = 10$ and $V = 100$ as illustrated below:

```
> limitTox <- 10.96
> doses <- c(12.6 34.655 44.69 60.807 83.689 100.371 )
> timeSampling <- seq(0,24,length.out=48)
> sigma <- rep(0.2,length(timeSampling))
> gen.scen <- sim.data(PKparameters=c(2,10,100),omegaIIV=0.7,omegaAlpha=0,
                       sigma,doses,limitTox,timeSampling,N=30, TR = 10)
```

Each trial's result is stored in a list of the R *"scen"* object, `gen.scen`, which regroups the subject's PK parameters, the concentration measurements for all patients and the simulated toxicities values at each dose level. Here, we provide some `sim.data` results for the first trial:

```
 Scenarios Settings (TR: 1)

Total number of patients in the trial: 30
The subject's PK parameters (ka, CL, V): 2 10 100
with a standard deviation of CL and V equals to: 0.7
The doses of the drug: 12.6 34.655 44.69 60.807 83.689 100.371
The real probabilities of toxicity: 0.001 0.05 0.1 0.2 0.35 0.45
Time after the drug administration (hours):
 0 0.511 1.021 1.532 2.043 2.553 3.064 3.574 4.085 4.596 5.106 5.617 6.128 6.638 7.149
 7.66 8.17 8.681 9.191 9.702 10.213 10.723 11.234 11.745 12.255 12.766 13.277 13.787
 14.298 14.809 15.319 15.83 16.34 16.851 17.362 17.872 18.383 18.894 19.404 19.915
 20.426 20.936 21.447 21.957 22.468 22.979 23.489 24
Threshold on the toxicity: 10.96
AUC with the sensitivity parameter
  [1]  0.985  2.710  3.495  4.756  6.546  7.850
  [7]  0.730  2.009  2.591  3.525  4.852  5.819
 [13]  1.453  3.996  5.153  7.012  9.650 11.574

                        ⋮

[175]  0.806  2.216  2.858  3.889  5.352  6.419
```

Based on the above AUC with the sensitivity parameter values and the chosen toxicity threshold $\tau_T = 10.96$, we observe for all patients (one row=one patient) in the second trial, the toxicity outcomes at each dose level (one dose=one column) as follows:

```
 Toxicity (0 indicates no toxicity and 1 toxicity) :

       dose 1 dose 2 dose 3 dose 4 dose 5 dose 6
pid 1       0      0      0      0      0      0
pid 2       0      0      0      0      0      0
pid 3       0      0      0      0      0      1
pid 4       0      0      0      0      1      1
pid 5       0      0      0      0      0      0

                        ⋮

       dose 1 dose 2 dose 3 dose 4 dose 5 dose 6
pid 25      0      0      0      0      1      1
pid 26      0      0      0      0      1      1
pid 27      0      0      1      1      1      1
pid 28      0      0      0      0      0      0
pid 29      0      0      0      0      0      0
pid 30      0      0      0      0      0      0

The PK parameter's estimations for each patient are:
```

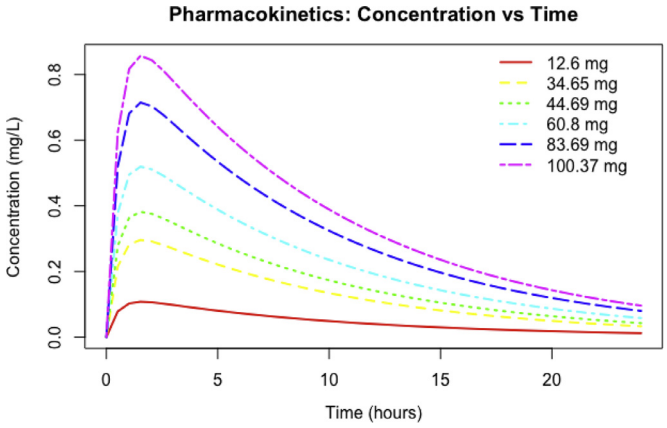


Fig. 3. Plot of the concentration of the drug vs time for 12.6, 34.65, 44.69, 60.8, 83.69 and 100.37 mg with $k_a = 2h^1$, $CL = 10Lh^1$ and $V = 100L$.

```
        ka      CL       V            ka      CL       V
pid 1    2 12.786 132.892    pid 16   2 15.655  93.047
pid 2    2 17.250 116.496    pid 17   2  9.101 131.441
pid 3    2  8.672  91.615    pid 18   2  6.373  27.317
pid 4    2  6.118  43.867    pid 19   2  9.106 102.490
pid 5    2 48.252  77.178    pid 20   2 11.275 186.811
pid 6    2 11.469 112.203    pid 21   2 10.829 171.712
pid 7    2 17.396 166.911    pid 22   2  5.623  65.968
pid 8    2  6.657  64.930    pid 23   2  8.360 727.193
pid 9    2 10.248 156.345    pid 24   2  4.526  95.767
pid 10   2  6.214  49.496    pid 25   2  6.854  55.886
pid 11   2 16.706 155.188    pid 26   2  5.699 116.239
pid 12   2  8.394  46.786    pid 27   2  3.736 113.431
pid 13   2 11.494  42.735    pid 28   2 12.682 249.079
pid 14   2 12.282 182.821    pid 29   2 48.888  78.613
pid 15   2  9.857  86.489    pid 30   2 15.636  93.866

> plot(gen.scen[[2]], lwd=2)
> legend(x=17, y=0.9, c("12.6 mg", "34.65 mg",
                "44.69 mg", "60.8 mg", "83.69 mg",
          "100.37 mg"),lty=c(1,2,3,4,5,6), lwd=2,
                      bty = "n", col=rainbow(6))
```

In this example, we used $\alpha = 1$ for all patients, but the user can simulate α from a log-normal distribution with mean=1 and standard deviation ω_α. Selecting $\omega_\alpha = 0$ implies $\alpha = 1$ as in this example.

Once again, providing the selected list of the generated *"scen"* object to the generic plot function `plot`, the user obtains a plot of the drug's concentration in the plasma against the time t. Under the same settings and outputs as above, the plot is implemented as follows:

Fig. 3 presents the PK concentration curves, described in Section 2, with the PK parameters $k_\alpha = 2h^{-1}$, $CL = 10Lh^{-1}$ and $V = 100L$ for the 6 doses (12.6, 34.65, 44.69, 60.8, 83.69 and 100.37 mg).

3.3. Dose-finding simulation (`nsim`)

The function `nsim` simulates a single or n prospective clinical trials. In the simulated trial, the dose is escalated stepwise cohort by cohort until the first toxicity response is observed and then the chosen dose-finding method design is applied (two-stage design) as suggested in Ursino et al. [8].

```
> nsim(doses, N, cohort, icon, theta, model, simulatedData, TR, prob = 0.9,
        AUCmethod = 2, options = list(nchains = 4, niter = 4000, nadapt = 0.8),
        betapriors = NULL, thetaL = NULL, p0 = 0, L = 0, CI = FALSE, seed = 190591)
```

In addition to the input arguments of `nextDose`, the `nsim` function has the following additional arguments: (i) `N`, the total sample size per trial, (ii) `cohort`, the cohort size, (iii) `TR`, the total number of trials to be simulated, (iv) `simulatedData`, a list for each trial containing previous simulated datasets as in 3.2.1, (v) `icon`, a vector containing the index of real blood sampling that enables the user to use all concentration points, previously simulated, or only a subset of them and (f) `AUCmethod`, a string number specifying the estimation method for AUC; valid choices are 1 for a "compartmental method" (see [8] for details) and 2 for non-compartmental method (defaults to 2). The estimated AUC for each patient is computed inside the function `nsim`, using only the concentration samples selected by `icon`. By default, all simulated samples are used. Similarly to the `nextDose` function, the user needs to choose one of the dose-finding models which are available in the package.

3.3.1. Demonstration

As an example, 10 trials (i.e. TR=10) were simulated with 30 patients (i.e. N=30) per trial, using the function `nsim`. At the beginning, the `simulatedData` is defined (in this case, we use the simulated data `gen.scen` generated in the Section 3.2.1), representing the true toxicity probabilities of each trial, where six dose levels are considered. The dose levels should be entered in a vector, as follows:

```
>  doses <- c(12.59972, 34.65492, 44.69007,
              60.80685, 83.68946, 100.37111)
```

The target toxicity probability `theta` is set to 0.2, meaning that the MTD is defined as the dose for which at most 20% of dose limiting toxicity (DLT) responses occur. For this example, the PKTOX method is selected and the 95% CI of the probability of toxicity at each dose is chosen to be estimated and included in the results since the input argument `CI` is set to TRUE. Since our simulation uses Stan models, we also need to specify the model's options, as a list, containing the number of chains, how many iterations each chain will use and the number of warm-up iterations. The default choice is:

```
> options <- list(nchains = 4, niter = 4000, nadapt = 0.8)
```

After setting the index of real blood sampling (i.e. `icon`) and entering the corresponding model's input parameters, we call the `nsim` function as following:

```
> icon <- c(2, 3, 4, 5, 6, 9, 19, 28, 38, 48)
> simResult <- nsim(doses, N=30, cohort=1, icon, theta=0.2, model="pktox",
                    simulatedData=gen.scen, TR=10, options=options, AUCmethod=1,
                    CI = TRUE)
```

The result is saved in the R object, named `simResult`, of a S4 class *"dosefinding"*. The output can be divided in three parts. The first part shows a data summary of the chosen dose-finding model (PKTOX in this case), the second part the Stan options and finally the dose-finding results including the percentage of the dose-allocation and the percentage of the MTD selection. The generated output is as follows:

```
    A. Data Summary (pktox model)
Number of simulations: 10
Total number of patients in the trial: 30
The time sampling: 0.511 1.021 1.532 2.043 2.553 4.085 9.191 13.787 18.894 24
Levels of Doses: 12.6 34.655 44.69 60.807 83.689 100.371
Concentration of the drug: 0.675 0.747 0.834 1.026 1.031 0.761 0.568 0.396 0.285 0.268

    B. STAN Model's Options
The Stan model runs with 4 MCMC chains which each chain has 4000 iterations
and 0.8 warmup iterations
```

Based on the above example, the next recommended dose level is 4 mg with a percentage of MTD selection equals to 60%.

```
C. Dose-Finding Results:

Dose                  STOP   1     2     3     4     5     6
Truth Probabilities   NA   0.001 0.05  0.10  0.20  0.35  0.45
Dose-Allocation (%)   NA   0.150 0.05  0.14  0.38  0.21  0.07
Selected % MTD       0.00 0.100 0.00  0.10  0.60  0.20  0.00

Recommendation is based on a target
toxicity probability of: 0.2
```

The MTD, toxicity responses as well as the dose escalation for each trial can be obtained as follows:

```
> MTD <- simResult@MTD
 [1] 4 4 5 5 1 3 4 4 4 4
> Toxicity <- simResult@toxicity
> DoseLevels <- simResult@doseLevels
```

The generic function `plot()` can be used in order to illustrate the dose escalation during the trial or the dose-toxicity response for each dose level. The main input argument is a *"dosefinding"* object. The simulation output can be shown graphically with the command:

```
> plot(simResult, TR=6, ask = TRUE, CI = TRUE)
Make a plot selection (or 0 to exit):

1: Plot trial summary
2: Boxplot sampling dose response
3: Plot posterior dose response with 95% CI

Selection:
```

where `TR` represents the number of trial (defaults to TR = 1) for which we want to plot the graph, `CI` indicates if the simulation's results include the estimated 95% of the probability of toxicity or not (defaults to `CI = TRUE`) and `ask` represents the plot selection index (defaults to ask = TRUE) showing a selection menu as above: the user should enter 1 to see the dose escalation allocation of the selected trial, 2 to create a boxplot of the sampling distribution of the probability of toxicity at each dose in the end of the trial over the total number of trials, and 3 to plot the final posterior distributions of the probability of toxicity at each dose (the plot includes the estimation along with the lines of the 95% CI for the selected trial). 0 is the command to exit and 2 is the default choice for the input `ask`. Note that, if the simulation's results don't include the 95% CI of probability of toxicity then the selection menu contains only the first two choices.

Fig. 4(A) shows the dose allocation plot based on our example, where the non-toxicity response is represented as a circle and the toxicity response as a cross. In addition, Fig. 4(B) and 4(C) present the output plot choosing, in the menu, 2 and 3, respectively. In Fig. 4(C), the 95% CI and prior probabilities of toxicity are represented as dotted and dashed lines, respectively. The red dot-dash line in the last two Figures represents the toxicity threshold which is used for the selection of the MTD.

Note that, since the Bayesian models are implemented in Stan, running simulations can take very long time. Moreover, simulations including the estimation of the 95% credible intervals (CI) for probability of toxicity at each dose level (i.e. `CI = TRUE`), can take more time than excluding them (i.e. `CI = FALSE`). In this case, to run the 10 trials including the 95% CI of probability of toxicity, about 30 min are needed on a single portable computer with an Intel Core i5. Instead, to simulate only the MTD under the same settings, without estimating the 95% CI, about 18 min are required on the same computer. A more expanded comparison of the `nsim` function and for 1,000 simulated trials, which is often used for the simulation studies in the literature of dose-finding methods, are shown in Appendix A. However, we suggest to run simulations on a dedicated server.

4. Conclusion

The `dfpk` package implements novel methods for dose-finding phase I clinical trials incorporating PK in the dose-toxicity relationships[8]. In this package, each method can be used during a prospective adaptive trial, where the dose for the next cohort of patients depends on the outcomes of the previous cohorts, in order to estimate the recommended dose for further clinical trials. It can also be used to perform simulations before the beginning of trial in order to study the robustness of the method to the different parameters setting choices. Running simulations is also useful to calibrate some parameters, but it takes time, therefore we suggest to run simulations on a dedicated server.

The package is user-friendly and several flexible inputs are allowed: for instance the user can generate by her/himself scenarios (simulated datasets) and pass them on to the function `nsim`, or change the hyper-prior parameters of the prior distributions used in the Bayesian regression. The package will also be updated according user suggestions and needs.

5. Discussion

Designing early phase clinical trials is of crucial importance, since all future steps in the clinical development or failures depend on these first results. Statistical computer programs, such as R, facilitate design and the checking of performance through simulations. An example of existing R packages can be found in [13]. However, to the best of our knowledge, there are no other software packages available that implement a formal integration of dose-finding and pharmacokinetics. Our R package can support interdisciplinary trial teams in implementing innovative dose-finding design using PK information in phase I studies.

Ursino et al. [8] compared a number of dose-finding methods under several scenarios, in order to verify their behaviour and characteristics. The PKCRM method behaves as the CRM alone when the L is very high. On the other hand, it gives the same probability of correct MTD selection (as the CRM) while reducing the probability of overdosing, when L is appropriately chosen. Therefore, this design is recommended when in preclinical phases non-monitorable toxicity has been observed or in some pediatric studies, when L can be easily set from a literature review. The PKLOGIT and PKTOX methods are recommended when more precise dose-response curve estimation is required. Compared to the CRM these methods are able to better estimate the probability of toxicity associated with each dose along with accurate MTD selection. In this way, a richer knowledge can be transmitted to subsequent phases of clinical development. The other methods have similar behaviour to any dose-toxicity regression, and can be used for comparisons in simulations.

The choice of the prior distributions is crucial. In this package, we set as default the prior distributions suggested in [8].

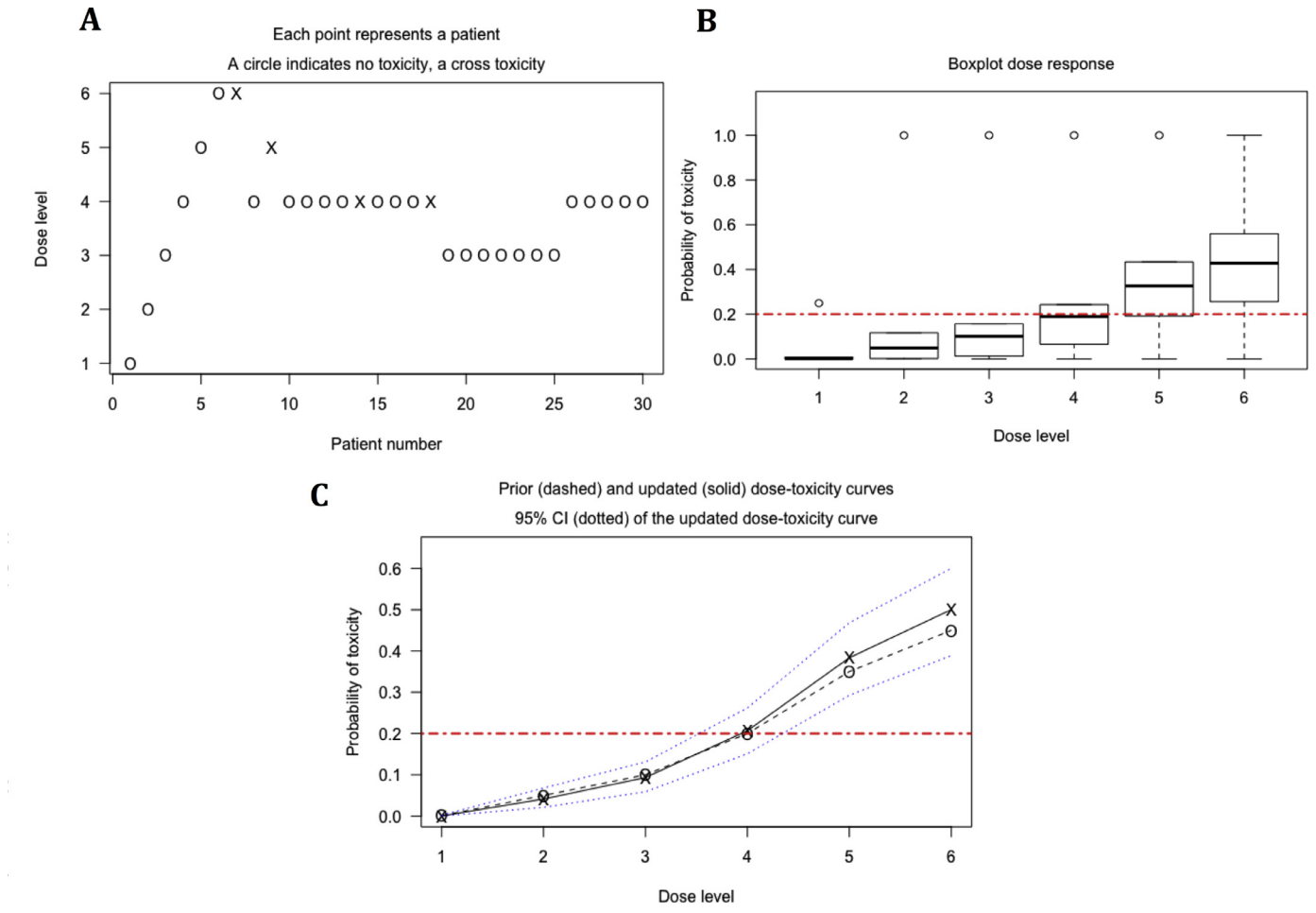


Fig. 4. (A) Plot of the dose escalation (for each patient) in the trial, (B) Boxplot of the sampling distribution of the probability of toxicity at each dose over the total number of trials, and (C) Plot of the posterior distributions of the probability of toxicity at each dose (including the estimation along with the lines of the 95% CI), according to the PKTOX method.

These prior hyperparameters were chosen after sensitivity analysis to give good performance in most cases. However, we suggest setting the prior hyperparamenters using preclinical information or other external pertinent information if this is available.

Acknowledgements

We thanks the three anonymous reviewers and the associated editor for their suggestions. This research was conducted as part of the European project, InSPiRe, Innovative Methodology for Small Populations Research. All the authors were funded by the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement number FP HEALTH 2013–602144.

Appendix A. Simulation times

Simulations based on Bayesian methods can be very tedious and time consuming. Therefore, for 1,000 simulated trials, we suggest to use a dedicated server. Moreover, running in parallel all the scenarios and not sequentially can also reduce the time of the simulations.

Table A.1 shows the estimated times for conducting the simulations of 10 and 1,000 trials, excluding the 95% credible intervals, for the methods PKTOX and PKCRM under several settings (i.e. the number of chains and iterations for the Bayesian algorithm).

Table A.1
Simulation's estimated times running 10 or 1,000 trials, in minutes and hours respectively, using different dose-finding methods under several settings for Bayesian algorithm (number of chains and iterations).

Bayesian settings	TR = 10		TR = 1,000	
	Methods		Methods	
	PKTOX	PKCRM	PKTOX	PKCRM
chains = 4, iter = 4000	18 min	10 min	≈ 32 h	≈ 17 h
chains = 4, iter = 6000	26 min	14 min	≈ 45 h	≈ 25 h
chains = 3, iter = 4000	13 min	7 min	≈ 23 h	≈ 13 h
chains = 6, iter = 4000	26 min	14.5 min	≈ 44 h	≈ 24 h

Appendix B. S4 classes

One of the big advantages of `dfpk` package is its flexible framework based on the S4 classes and methods structure. S4 classes have allowed us to construct rich and complicated data representations that nevertheless seem simple to the end user. The class is the abstract definition, while every time we actually use it to store the results for a given data set, we create an object of the class.

Three S4 classes are available, the `dose-class`, the `scen-class` and the `dosefinding-class`, in order to store, show or plot the corresponding results of the main R functions `nextDose`, `sim.data` and `nsim`, respectively. You can click on the corresponding help pages as background information for the next steps.

Table B.1
The required slots (i.e. arguments) in the `dose` class.

N	The total number of enrolled patients.
y	A binary vector of toxicity outcomes from previous patients; 1 indicates a toxicity, 0 otherwise.
AUCs	A vector with the computed AUC values of each patient.
doses	A vector with the doses panel.
x	A vector with the dose levels assigned to the patients in the trial.
theta	The toxicity target.
options	A list of Stan model's options.
newDose	The next recommended dose (RD) level; equals to 0 if the trial has stopped, according to the stopping rules.
pstim	The estimated mean probabilities of toxicity.
pstimQ1	The 1st quartile of estimated probability of toxicity.
pstimQ3	The 3rd quartile of estimated probability of toxicity.
parameters	The Stan model's estimated parameters.
model	A character string to specify the selected dose-finding model used in the method.

In this section, a briefly description of the accessible classes is shown.

1. `dose-class`

The `dose-class` is created to store and present the next recommended dose level in an ongoing trial through the R function `nextDose`. We can look in detail at the structure of the class as follows:

```
> setClass("dose", slots = list(N = "numeric", y = "numeric", AUCs = "numeric",
        doses ="numeric", x = "numeric", theta = "numeric", options = "list",
        newDose="ClassNewDose", pstim="numeric", pstimQ1="ClassNewDose",
        pstimQ3="ClassNewDose", parameters="numeric", model = "character"))
```

where, `ClassNewDose` is a union of classes "numeric", "logical" and "NULL".

Accordingly to the structure, the `dose` class consists of 13 slots (i.e. arguments) which each one has a specific type. Table B.1 gives a brief definition of each corresponding slots in the `dose` class.

An object that comes from the `dose-class` must contain all the above slots. The slots are accessed using @, just as components of a list that are accessed using $. Here, an illustration of how to access the slots of an object is given.

We suppose that `nextD` is an object of the `dose` class. For example, the slots `model` and `y` can be obtained as follows:

```
> nextD@model
[1] "pktox"
> nextD@y
 [1] 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0
...
```

where, PKTOX was the selected dose-finding model and the vector (0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0) was the toxicity outcome for each patient that are used in the function `nextDose`.

Once the classes are defined, we probably want to perform some computations on objects. In most cases we do not care how the object is stored internally, the computer should decide how to perform the tasks. The S4 way of reaching this goal is to use generic functions and method dispatch.

Based on our above example, we hypothesised that we stored the output of the function `nextDose` in the object `nextD`. To present the results we can use either the method `show` or `print` as follows:

```
> print(nextD)
> show(nextD)
```

Both methods give a nice and simple presentation of the outcome that is stored in the object `nextD`. In any case where user wants to change how the results are presented, she/he can easily do it by setting her/his own `show` or `print` method in the class `dose`. Similarly, a `plot` generic function is defined for this class and can be easily accessed through the command:

```
plot(nextD)
```

2. `scen-class`

Table B.2
The required slots (i.e. arguments) in the `scen` class.

PKparameters	Subject's pharmacokinetic's (PK) parameters from the population distributions defined by the population mean.
nPK	The length of time points.
time	The sampling time points.
idtr	The id number of the corresponding simulated dataset.
N	The total sample size per trial.
doses	A vector of the doses panel.
preal	The prior toxicity probabilities.
limitTox	The toxicity threshold.
omegaIIV	The inter-individual variability for the clearance and the volume of distribution.
omegaAlpha	The patient's sensitivity parameter.
conc	The concentration computed at the PK population values.
concPred	The concentration values with proportional errors for each patient at each dose.
tox	The toxicity outcome.
tab	A summary matrix containing the sampling time points at the first row followed by `concPred`, `parameters` and `alphaAUC`. It used by the simulation function `nsim`.
parameters	The simulated PK parameters of each patient.
alphaAUC	A vector with the computed AUC values of each patient.

A `scen` is a S4 class to save and show a dataset simulated by the function `sim.data`. We can look in detail at the structure of the class `scen` as follows:

```
> setClass("scen", slots = list(PKparameters="numeric", nPK="numeric",
        time="numeric", idtr = "numeric", N = "numeric", doses="numeric",
        preal = "numeric", limitTox="numeric", omegaIIV="numeric",
        omegaAlpha="numeric", conc="matrix", concPred="numeric",
        tox="matrix", tab="matrix", parameters="matrix", alphaAUC="numeric"))
```

Thanks to S4 classes, the user can easily create his/her own datasets. For example, he/she can create a new object for each simulated trial, named UserData, and store it in a `scen-class` by a similar way using the command `new` in the following way:

```
>  UserData <- new("scen", PKparameters, nPK, time, idtr, N, doses, preal,
        limitTox, omegaIIV, omegaAlpha, conc, concPred, tox, tab,
        parameters, alphaAUC)
```

and then add it in a list, along with the other simulated datasets.

A detailed definition of each slot is presented in the Table B.2.

Once again, user can access to the slots and apply the generic functions and methods in this class by exactly the same way as in `dose-class`. Assume that we run 10 (i.e. TR=10) simulated datasets using the function `sim.data` and `simulatedData` is a `scen` object then:

```
> idtr = 2    # for example select the second simulated dataset
> simulatedData[[idtr]]@PKparameters
> show(simulatedData[[idtr]])
> plot(simulatedData[[idtr]])
...
```

3. `dosefinding-class`

Lastly, a third S4 class is available in the package dfpk, called `dosefinding`, which is created to store all the dose-finding results that are simulated through the R function `nsim`.

Table B.3
The required slots (i.e. arguments) in the `dosefinding` class.

pid	Patient's ID provided in the study.
N	The total sample size per trial.
time	The sampling time points.
doses	A vector with the doses panel.
conc	The estimated concentration values for each patient at each dose.
p0	The skeleton of CRM for PKCRM.
L	The AUC threshold to be set before starting the trial for PKCRM.
nchains	The number of chains for the Stan model.
niter	The number of iterations for the Stan model.
nadapt	The number of warmup iterations for the Stan model.
newDose	The next maximum tolerated dose (MTD) if TR=1 otherwise the percentage of MTD selection for each dose level after all trials starting from dose 0; equals to 0 if the trial has stopped before the end, according to the stopping rules.
MTD	A vector containing the next maximum tolerated doses (MTD) of each trial (TR); equals to 0 if the trial has stopped before the end, according to the stopping rules.
MtD	The final next maximum tolerated (MTD) dose after all the trials.
theta	The toxicity threshold.
doseLevels	A vector of dose levels assigned to patients in the trial.
toxicity	The estimated toxicity outcome.
AUCs	A vector with the computed AUC values of each patient.
TR	The total number of trials to be simulated.
preal	The prior toxicity probabilities.
pstim	The estimated mean probabilities of toxicity.
pstimQ1	The 1st quartile of the estimated probability of toxicity.
pstimQ3	The 3rd quartile of the estimated probability of toxicity.
model	A character string to specify the selected dose-finding model used in the method.
seed	The seed of the random number generator that is used at the beginning of each trial.

It's structure is given below:

```
> setClass("dosefinding", slots = list(pid="numeric", N ="numeric",
        time="numeric", doses = "numeric", conc="numeric",
        p0 = "numeric", L = "numeric",  nchains = "numeric",
        niter = "numeric", nadapt ="numeric", newDose = "ClassNewDose",
        MTD = "ClassNewDose", MtD = "numeric", theta = "numeric",
        doseLevels="matrix", toxicity= "matrix", AUCs="matrix", TR="numeric",
        preal = "numeric", pstim = "list", pstimQ1 = "list", pstimQ3 = "list",
        model = "character", seed= "matrix"))
```

where, 21 different slots are available. Table B.3 defines all the possible slots.

Identically with the `dose` and `scen` S4 classes, the slots can picked using the @ "operator" and the generic functions and methods can be applied in the same way.

References

[1] S. Chevret, Statistical Methods for Dose-Finding Experiments of Statistics in Practice, John Wiley and Sons, Chichester, West Sussex, England, 2006.

[2] H. Derendorf, L.J. Lesko, P. Chaikin, W.A. Colburn, P. Lee, R. Miller, R. Powell, G. Rhodes, D. Stanski, J. Venitz, Pharmacokinetic/pharmacodynamic modeling in drug research and development, J. Clin. Pharmacol. 40 (12) (2000) 1399–1418.

[3] E. Comets, S. Zohar, A survey of the way pharmacokinetics are reported in published phase i clinical trials, with an emphasis on oncology, Clin. Pharmacokinet. 48 (6) (2009) 387–395.

[4] F. Bretz, J.C. Pinheiro, M. Branson, Combining multiple comparisons and modeling techniques in dose-response studies, Biometrics 61 (3) (2005) 738–748.

[5] S. Piantadosi, G. Liu, Improved designs for dose escalation studies using pharmacokinetic measurements, Stat. Med. 15 (15) (1996) 1605–1618, doi:10.1002/(SICI)1097-0258(19960815)15:15⟨1605::AID-SIM325⟩3.0.CO;2-2.

[6] S. Patterson, S. Francis, M. Ireson, D. Webber, J. Whitehead, A novel Bayesian decision procedure for early-phase dose-finding studies, J. Biopharm. Stat. 9 (4) (1999) 583–597, doi:10.1081/BIP-100101197.

[7] J. Whitehead, Y. Zhou, L. Hampson, E. Ledent, A. Pereira, A bayesian approach for dose-escalation in a phase i clinical trial incorporating pharmacodynamic endpoints, J. Biopharm. Stat. 17 (6) (2007) 1117–1129. PMID: 18027220 doi: 10.1080/10543400701645165.

[8] M. Ursino, S. Zohar, F. Lentz, C. Alberti, T. Friede, N. Stallard, E. Comets, Dose-finding methods for phase I clinical trials using pharmacokinetics in small populations, Biom. J. (2017), doi:10.1002/bimj.201600084.

[9] R Core Team, R: a language and environment for statistical computing, in: R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0, http://www.R-project.org/.

[10] J. Whitehead, S. Patterson, D. Webber, S. Francis, Y. Zhou, Easy-to-implement Bayesian methods for dose-escalation studies in healthy volunteers, Biostatistics 2 (1) (2001) 47–61, doi:10.1093/biostatistics/2.1.47.

[11] J. O'Quigley, M. Pepe, L. Fisher, Continual reassessment method: a practical design for phase 1 clinical trials in cancer, Biometrics 46 (1) (1990) 33–48.

[12] G. Lestini, C. Dumont, F. Mentré, Influence of the size of cohorts in adaptive design for nonlinear mixed effects models: an evaluation by simulation for a pharmacokinetic and pharmacodynamic model for a biomarker in oncology, Pharm. Res. 32 (10) (2015) 3159–3169, doi:10.1007/s11095-015-1693-3.

[13] E. Zhang, H.G. Zhang, Cran task view: clinical trial design, monitoring, and analysis, 2016, (https://cran.r-project.org/web/views/ClinicalTrials.html). Accessed: 2017-10-22.