Journal of Statistical Software

March 2023, Volume 106, Issue 4. *doi: 10.18637/jss.v106.i04*

drda: An R Package for Dose-Response Data Analysis Using Logistic Functions

Alina Malyutina
University of Helsinki

Jing Tang
University of Helsinki

Alberto Pessia
University of Helsinki

Abstract

Analysis of dose-response data is an important step in many scientific disciplines, including but not limited to pharmacology, toxicology, and epidemiology. The R package **drda** is designed to facilitate the analysis of dose-response data by implementing efficient and accurate functions with a familiar interface. With **drda** it is possible to fit models by the method of least squares, perform goodness-of-fit tests, and conduct model selection. Compared to other similar packages, **drda** provides in general more accurate estimates in the least-squares sense. This result is achieved by a smart choice of the starting point in the optimization algorithm and by implementing the Newton method with a trust region with analytical gradients and Hessian matrices. In this article, **drda** is presented through the description of its methodological components and examples of its user-friendly functions. Performance is evaluated using both synthetic data and a real, large-scale drug sensitivity screening dataset.

Keywords: curve fitting, dose-response, drug sensitivity, logistic function, nonlinear regression.

1. Introduction

Inferring dose-response relationships is indispensable in many scientific disciplines. In cancer research, for example, estimating the magnitude of a chemical compound effect on cancer cells holds substantial promise for clinical applications. The dose-response relationship can be modeled via a nonlinear parametric function expressed as a dose-response curve. The choice of the curve is achieved by selecting the parameter values that lead to the best fit of the observations. Since conclusions about compound efficacy, for example its potency to kill the cells, are based on the estimated dose-response curve, it is of great importance to determine the curve parameters as accurately as possible.

Currently, there are multiple R (R Core Team 2022) packages that provide tools for dose-response fitting, such as **drc** (Ritz, Baty, Streibig, and Gerhard 2015; Ritz and Strebig 2016), **nplr** (Commo and Bot 2016), and **DoseFinding** (Bornkamp, Pinheiro, and Bretz 2019). The **drc** package contains various functions for nonlinear regression analysis of biological assays. The package accepts continuous, binary and discrete responses as quantification of biological effect. It allows the user to choose a nonlinear model for the dose-response curve fitting from a wide spectrum of sigmoidal functions, which are normally used to capture the dose-response relationship as their S-shape is in line with empirical observations from experiments. In the **drc** package, a user can specify initial model parameters to facilitate the optimization process or rely on the default starter functions. The package also enables the user to assign weights to the observations to adjust for possible variance heterogeneity in the response values. Parameter estimation in **drc** is done by maximum likelihood, which simplifies to the nonlinear least squares method under the assumption of response normality.

In contrast to **drc**, package **nplr** focuses only on logistic models and does not allow to select the data distribution. As a feature, the package facilitates the choice of observation weights via implementing three options: residual-based, standard (or within-replicate variance-based), and general, which utilizes the fitted response values. Additionally, the package provides confidence intervals on the predicted doses and the trapezoid and the Simpson's rule (Abramowitz and Stegun 1965, Chapter 25) to evaluate the area under the curve.

The **DoseFinding** package provides more flexibility than **drc** and **nplr**. It allows for the fitting of multiple linear and nonlinear dose-response models and to design dose-finding experiments. Similarly to **drc**, it provides several options for the data distribution, but by default it uses an assumption of normality with equal variance. Compared to **drc** and **nplr**, package **DoseFinding** utilizes a grid search as a starting point selection method in case the user did not specify their own. It also applies boundaries to parameters of a nonlinear model either specified by the user or through internal default settings.

To find the optimal solution in a p-dimensional space, where p is the number of parameters, all packages apply iterative Newton methods, which are widely used numerical procedures for finding stationary points of a differentiable function (Nocedal and Wright 2006). The **drc** package directly calls the R function `optim()` implementing the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method (Fletcher 2000) for unconstrained optimization, or limited-memory BFGS (L-BFGS-B), which handles simple box constraints for constrained optimization (Liu and Nocedal 1989). These two methods are quasi-Newton methods, which are frequently used in cases when the function derivatives are not feasible or too complicated to obtain, as they utilize numerical approximations of the function's Hessian matrix. In contrast, the **nplr** package relies on the `nlm()` function, which uses the classic Newton approach. By default, both the gradient and Hessian are approximated numerically, however the user can also provide the first and second analytical derivatives. The **DoseFinding** package applies different optimization routines depending on the models of choice. For sigmoidal models, which have linear and nonlinear function parameters, the package performs numerical optimization just for nonlinear ones, while optimizing the linear parameters in every iteration of the algorithm. At its core, **DoseFinding** applies the R function `nlminb()`, using a quasi-Newton algorithm similar to the BFGS method utilized by **drc**.

While all packages have been extremely helpful with a wide range of real applications, we found that they often present inconsistent results when fitting the same logistic model on the same data. We introduce here the R package **drda** (Pessia 2022), which provides a novel and more accurate dose-response data analysis using logistic curves via: (i) establishing a smart initialization procedure to increase the chances of converging to the global solution; (ii) applying a more advanced Newton method with a trust region and (iii) relying on analytical gradient and Hessian formulas instead of numerical approximations in problematic cases to assure proper convergence; (iv) computing confidence intervals for the whole dose-response curve using multiple comparisons test correction. Besides, similar to other packages, **drda** provides tools to compare the fitted curve against a linear model or other logistic models, to compute confidence intervals for the estimated parameters, and to plot multiple models in a user-friendly manner.

The most important feature of any optimization routine remains the closeness of its solution to the true optimizer of the objective function, such as the maximum likelihood estimate. One of the main disadvantages when it comes to numerical optimization is the possibility of converging to a local optimum instead of the correct answer we seek. This situation can easily happen when either the function is not well approximated by a quadratic shape in a neighborhood of the current candidate solution, or when the starting point is far from the global optimum (either the algorithm is not able to converge in a reasonable number of steps or it simply converges to a wrong solution). To cope with such scenarios, we implement here the Newton method with a trust region (Steihaug 1983), which has been shown to be a robust optimization technique for mitigating issues usually encountered in optimization problems. The method is more stable than other Newton-based methods, especially for cases when it is problematic to approximate a function with a quadratic surface (Sorensen 1982). Additionally, **drda** uses a multi-step initialization algorithm in order to ensure the right direction in the optimization routine. With our strategy, **drda** is able to find the true least squares estimate in problematic cases where the **drc**, **nplr**, and **DoseFinding** packages instead fail.

Once the least-squares estimate is found, **drda** provides the user with routines for assessing goodness-of-fit and reliability of the estimates. Assuming a Gaussian distribution with equal variance for the observed data, it is possible to compare the fitted model against, for example, a flat horizontal line or a logistic model with a different number of parameters. The **drda** package provides the likelihood-ratio test (LRT), the Akaike information criterion (AIC, Akaike 1974), and the Bayesian information criterion (BIC, Schwarz 1978) as a way to compare the goodness-of-fit of competing models.

The paper is organized as follows: We first describe the methodological components of **drda** in Section 2; show how the package is implemented in Section 3 (including practical examples in Section 3.2); and provide a comparison of **drda** against packages **drc**, **nplr**, and **DoseFinding** using simulations and a high-throughput dose-response dataset in Section 4. We conclude the article with a summary and discussion in Section 5.

2. Methodological framework

2.1. Generalized logistic and log-logistic function

Package **drda** implements the generalized logistic and log-logistic functions as the core models for fitting dose-response data. The generalized logistic function, also known as Richards' curve (Richards 1959), is the 5-parameter function

$$f(x; \boldsymbol{\psi}) = \alpha + \delta \left(1 + \nu \exp\{-\eta(x - \phi)\}\right)^{-1/\nu}, \tag{1}$$

solution to the differential equation

$$\frac{\partial}{\partial x} f(x;\boldsymbol{\psi}) = \frac{\eta}{\nu}\left(1 - \left(\frac{f(x;\boldsymbol{\psi}) - \alpha}{\delta}\right)^{\nu}\right)\left(f(x;\boldsymbol{\psi}) - \alpha\right),$$

where $\boldsymbol{\psi} = (\alpha, \delta, \eta, \phi, \nu)^\top$ and $x \in \mathbb{R}$.

Throughout this article, and in our package, we will use the convention $\eta \geq 0$ to avoid identifiability issues. For example, when $\nu = 1$, it holds that $f(x; \alpha, \delta, \eta, \phi, 1) = f(x; \alpha + \delta, -\delta, -\eta, \phi, 1)$. Furthermore, to have a sigmoidal curve, a common requirement in dose-response data analysis, we will also assume that $\nu \geq 0$. When $\nu < 0$, in fact, the curve is unbounded or even complex.

Our constraints have the benefit of giving the five parameters a clear and easy interpretation: α is the value of the function when x approaches negative infinity, δ is the (signed) height of the curve, that is $\alpha + \delta$ is the value of the function when x approaches positive infinity, η is the steepness or growth rate of the curve, ϕ is related to the mid-value of the function, and ν regulates at which asymptote is the curve maximum growth. In dose-response data analysis, parameter δ represents the maximum effect attributable to the drug and it is often denoted in the literature as E_{max} (Macdougall 2006). Refer to Figure 1 for a visual explanation of the five parameters.

When $\nu = 1$ we obtain the 4-parameter logistic function. If we also set $(\alpha, \delta)^\top = (0, 1)^\top$ or $(\alpha, \delta)^\top = (1, -1)^\top$ we obtain the 2-parameter logistic function. When $\nu = 1$ the parameter ϕ represents the value at which the function is equal to its midpoint, that is $\alpha + \delta/2$. In such a case, as a measure of drug potency, ϕ is also known as the half maximal effective log-concentration or log-EC$_{50}$. As a measure of antagonist drug potency, ϕ is also known as the half maximal inhibitory log-concentration (log-IC$_{50}$). When $\nu \to 0$ we obtain the Gompertz function, i.e.,

$$\lim_{\nu \to 0} f(x;\boldsymbol{\psi}) = \alpha + \delta \exp\left\{-\exp\{-\eta(x - \phi)\}\right\}.$$

The domain of the logistic function is the whole real line, that is $x \in \mathbb{R}$. In dose-response data analysis, this means that the logistic function is a model for mapping log-doses to responses. When data is provided on the actual dose-scale, we can use the log-logistic function instead:

$$f_+(d;\boldsymbol{\psi}) = \alpha + \delta\left(\frac{d^\eta}{d^\eta + \nu\phi_+^\eta}\right)^{\frac{1}{\nu}}, \tag{2}$$

where $d = e^x$ and $\phi_+ = e^\phi$. Interpretation of parameters is equivalent to the logistic function with the only exception of ϕ_+ being now defined on the positive real line. Note that the log-logistic function is well-defined also for a dose of 0, where the function is equal to α. In the literature, the log-logistic function with $\nu = 1$ is also referred to as the E_{max} model (Macdougall 2006).

2.2. Normal nonlinear regression

For a particular dose d_k $(k = 1, \ldots, m)$ let $(y_{ki}, w_{ki})^\top$ represent respectively the ith observed outcome $(i = 1, \ldots, n_k)$ and its associated positive weight. If observations have all the same importance, we simply set $w_{ki} = 1$ for all k and i. We assume that each unit has expected

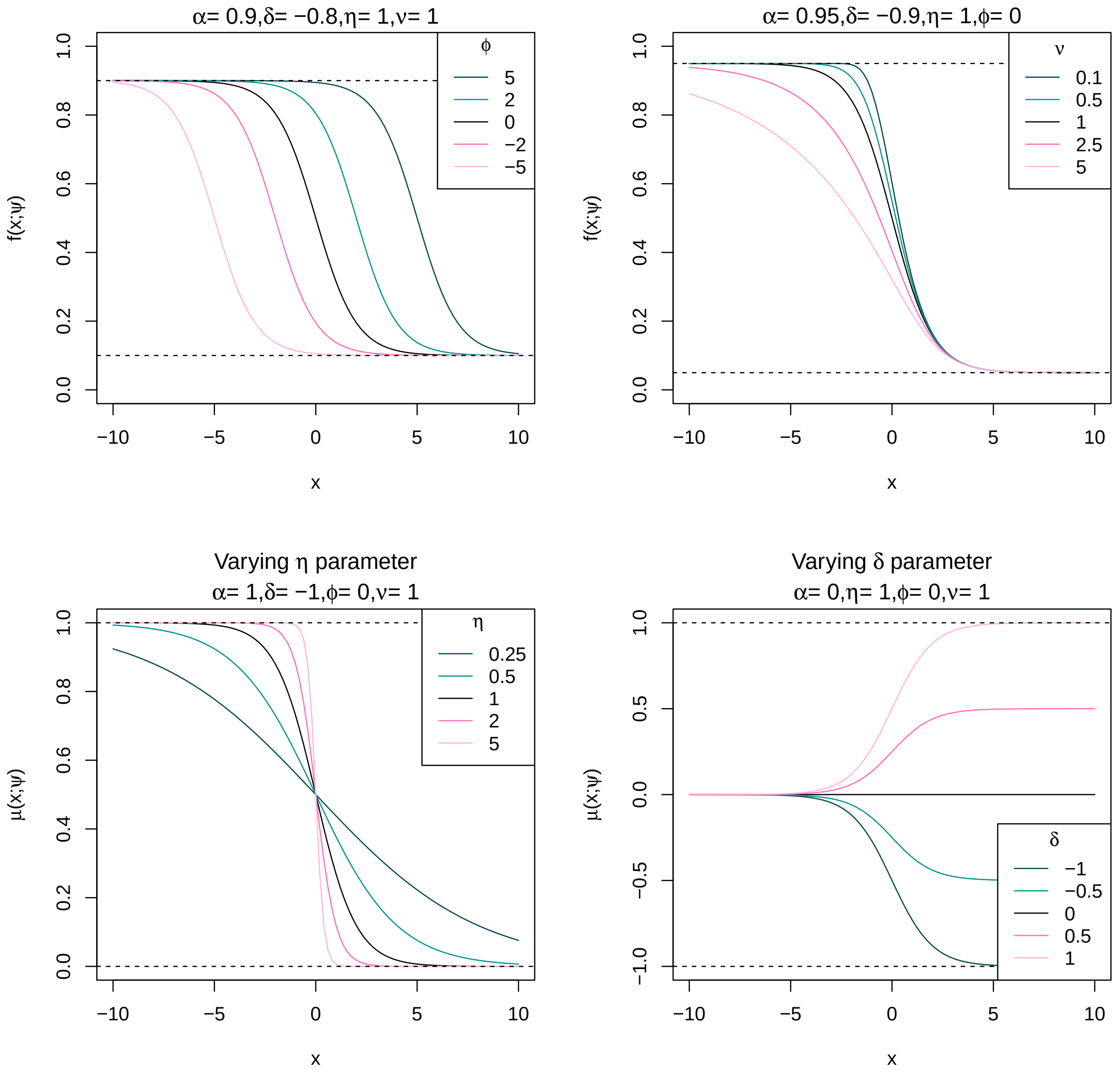

Figure 1: Generalized (5-parameter) logistic function with various choices of parameters.

value and variance

$$\mathbb{E}[Y_{ki}|d_k, \boldsymbol{\psi}] = \mu(d_k; \boldsymbol{\psi}),$$
$$\mathbb{V}[Y_{ki}|w_{ki}, \sigma] = \frac{\sigma^2}{w_{ki}},$$

where $\mu(d_k; \boldsymbol{\psi})$ is a nonlinear function of the dose d_k and a vector of unknown parameters $\boldsymbol{\psi}$. Parameter $\sigma > 0$ is instead the standard deviation common to all observations. In our package, $\mu(d_k; \boldsymbol{\psi})$ is simply the generalized log-logistic function (Equation 2) or the generalized logistic function (Equation 1) with the transformation $d_k = e^{x_k}$.

By assuming the observations to be stochastically independent and normally distributed, the joint log-likelihood function is

$$l(\boldsymbol{\psi}, \sigma) = -\frac{1}{2}\Bigg(n \log(2\pi) + n \log(\sigma^2) - \sum_{k=1}^{m}\sum_{i=1}^{n_k} \log(w_{ki}) + \frac{1}{\sigma^2}\sum_{k=1}^{m}\sum_{i=1}^{n_k} w_{ki}(y_{ki} - \bar{y}_k)^2 + \\ + \frac{1}{\sigma^2}\sum_{k=1}^{m} w_{k.}(\bar{y}_k - \mu(d_k; \boldsymbol{\psi}))^2\Bigg),$$

where n_k is the sample size at dose k, $n = \sum_k n_k$ is the total sample size, $\bar{y}_k = (\sum_i w_{ki} y_{ki})/w_{k.}$ is the weighted average corresponding to dose d_k and $w_{k.} = \sum_i w_{ki}$. The maximum likelihood estimate $\hat{\boldsymbol{\psi}}$ is obtained by minimizing the residual sum of squares from the means, i.e.,

$$\hat{\boldsymbol{\psi}} = \arg\min_{\psi \in \Psi} \frac{1}{2}\sum_{k=1}^{m} w_{k.}(\bar{y}_k - \mu(d_k; \boldsymbol{\psi}))^2 = \arg\min_{\psi \in \Psi} g(\boldsymbol{\psi}). \tag{3}$$

The maximum likelihood estimate of the variance is

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{k=1}^{m}\sum_{i=1}^{n_k} w_{ki}(y_{ki} - \mu(d_k; \hat{\boldsymbol{\psi}}))^2 = \frac{D^2}{n},$$

while its unbiased estimate is

$$s^2 = \frac{D^2}{n - p},$$

where p is the total number of parameters estimated from the data.

For convenience we will use from now on the simplified notation μ_k to denote the function $\mu(d_k; \boldsymbol{\psi})$. It is important to remember that μ_k will always be a function of a dose d_k and a particular parameter value $\boldsymbol{\psi}$. We will also use the notation $g^{(s)}$ and $g^{(st)}$ to denote respectively the first- and second-order partial derivatives of function $g(\boldsymbol{\psi})$, with respect first to ψ_s and then ψ_t.

Partial derivatives of the sum of squares $g(\boldsymbol{\psi})$ are

$$\begin{aligned} g^{(s)} &= \sum_{k=1}^{m} w_{k.}(\mu_k - \bar{y}_k)\mu_k^{(s)}, \\ g^{(st)} &= \sum_{k=1}^{m} w_{k.}\left((\mu_k - \bar{y}_k)\mu_k^{(st)} + \mu_k^{(s)}\mu_k^{(t)}\right). \end{aligned}$$

The gradient and Hessian of $g(\boldsymbol{\psi})$ are therefore

$$\begin{aligned} \nabla_{\boldsymbol{\psi}} g &= \sum_{k=1}^{m} w_{k.}(\mu_k - \bar{y}_k)\nabla_{\boldsymbol{\psi}}\mu_k, \\ \mathbf{H}_{\boldsymbol{\psi}} g &= \sum_{k=1}^{m} w_{k.}\left((\mu_k - \bar{y}_k)\mathbf{H}_{\boldsymbol{\psi}}\mu_k + (\nabla_{\boldsymbol{\psi}}\mu_k)(\nabla_{\boldsymbol{\psi}}\mu_k)^\top\right). \end{aligned}$$

From the previous expressions we can easily retrieve the observed Fisher information matrix, that is the negative Hessian matrix of the log-likelihood evaluated at the maximum likelihood estimate, as

$$\mathbf{I}(\boldsymbol{\psi}, \sigma) = \frac{1}{\sigma^2}\begin{pmatrix} \mathbf{H}_{\boldsymbol{\psi}} g & -2\nabla_{\boldsymbol{\psi}} g/\sigma \\ -2(\nabla_{\boldsymbol{\psi}} g)^\top/\sigma & q \end{pmatrix}, \tag{4}$$

where

$$q = \frac{3 \sum_k \sum_i w_{ki}(y_{ki} - \mu_k)^2}{\sigma^2} - n.$$

It is also worth noting that the (expected) Fisher information matrix is

$$\mathcal{I}(\boldsymbol{\psi}, \sigma) = \frac{1}{\sigma^2} \begin{pmatrix} \sum_k w_{k.} \left(\nabla_{\boldsymbol{\psi}} \mu_k\right) \left(\nabla_{\boldsymbol{\psi}} \mu_k\right)^\top & \mathbf{0} \\ \mathbf{0} & 3 \sum_k \sum_i w_{ki} - n \end{pmatrix}. \tag{5}$$

2.3. Statistical inference

When closed-form solutions of maximum likelihood estimates are missing, also closed-form expressions of other inferential quantities are not available. Fortunately, we can still rely on asymptotic, large sample size considerations, to obtain approximate values of quantities of interest. Obviously, the larger the sample size the better the approximation.

Using either versions (Equation 4 or Equation 5) of the Fisher information matrix we can calculate approximate confidence intervals. In fact, we can think of the Fisher information matrix as an approximate precision matrix, so that we only have to invert the matrix and take diagonal elements as approximate variance estimates. In our package we use the observed Fisher information matrix (Equation 4) because it is shown to perform better with finite sample sizes (Efron and Hinkley 1978). As an example, an approximate confidence interval for generic parameter ψ_j is

$$\hat{\psi}_j \pm t_{n-p,\alpha} \sqrt{\left(I(\hat{\boldsymbol{\psi}}, \hat{\sigma})^{-1}\right)_{jj}},$$

where $t_{n-p,\alpha}$ is the appropriate quantile of level α of a Student's t-distribution with $n - p$ degrees of freedom and $\left(I(\hat{\boldsymbol{\psi}}, \hat{\sigma})^{-1}\right)_{jj}$ is the jth element in the diagonal of the inverse observed Fisher information matrix. Using the delta method (Oehlert 1992) we can compute approximate point-wise confidence intervals for the mean function

$$\mu(d_k; \hat{\boldsymbol{\psi}}) \pm t_{n-p,\alpha} \sqrt{s^2 \left(\nabla_{\hat{\boldsymbol{\psi}}} \mu_k\right)^\top \left(\mathbf{H}_{\hat{\boldsymbol{\psi}}} f\right)^{-1} \left(\nabla_{\hat{\boldsymbol{\psi}}} \mu_k\right)}$$

or for a new, yet to be observed, value $y(d)$

$$\mu(d; \hat{\boldsymbol{\psi}}) \pm t_{n-p,\alpha} \sqrt{s^2 \left(1 + \left(\nabla_{\hat{\boldsymbol{\psi}}} \mu\right)^\top \left(\mathbf{H}_{\hat{\boldsymbol{\psi}}} f\right)^{-1} \left(\nabla_{\hat{\boldsymbol{\psi}}} \mu\right)\right)}.$$

We can also construct a (conservative and approximate) confidence band over the whole mean function $\mu(\cdot; \boldsymbol{\psi})$ with the correction proposed by Gsteiger, Bretz, and Liu (2011):

$$\mu(d; \hat{\boldsymbol{\psi}}) \pm \sqrt{q_{p,\alpha} s^2 \left(\nabla_{\hat{\boldsymbol{\psi}}} \mu\right)^\top \left(\mathbf{H}_{\hat{\boldsymbol{\psi}}} f\right)^{-1} \left(\nabla_{\hat{\boldsymbol{\psi}}} \mu\right)},$$

where $q_{p,\alpha}$ is the appropriate quantile of level α of a χ^2-distribution with p degrees of freedom.

2.4. Optimization by the Newton method with a trust region

A closed-form formula of the maximum likelihood estimate $\hat{\boldsymbol{\psi}}$, that is the solution of Equation 3, is in general not available for nonlinear regression models. We can, however, try to minimize numerically the sum of squares $g(\boldsymbol{\psi})$.

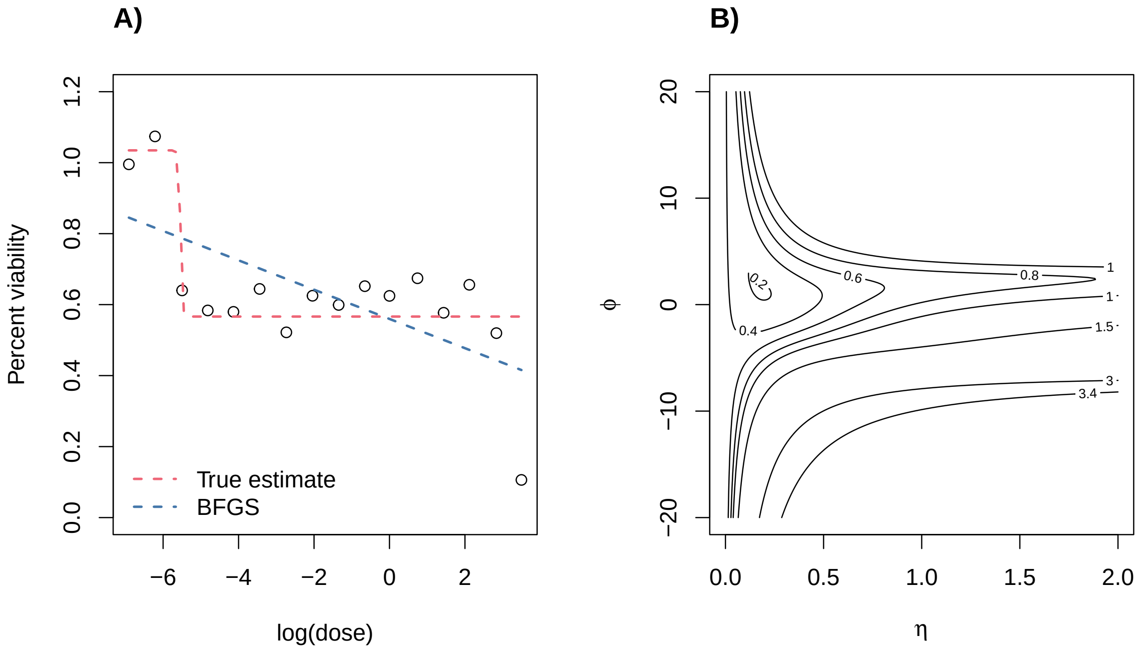

Figure 2: Problematic real data (cell line: BT-20, compound: BI-2536, dataset: CTRPv2) (Rees *et al.* 2016; Seashore-Ludlow *et al.* 2015; Basu *et al.* 2013). A) 4-parameter logistic function as fitted by the BFGS algorithm. Starting point $\boldsymbol{\psi} = (\alpha, \delta, \eta, \phi)^\top = (1, -1, 1, 0)^\top$. B) Contour plot of the residual sum of squares $g(\boldsymbol{\psi})$ with respect to parameters η and ϕ. Fixed parameters $\alpha = 1$ and $\delta = -1$.

Suppose that our algorithm is at iteration t with current solution $\boldsymbol{\psi}_t$. We want to find a new step u such that $g(\boldsymbol{\psi}_t + u) < g(\boldsymbol{\psi}_t)$. We start by illustrating the standard Newton method. We approximate our function by a second-order Taylor expansion, that is

$$g(\boldsymbol{\psi}_t + u) \approx g(\boldsymbol{\psi}_t) + \nabla_{\psi_t}^\top u + \frac{1}{2} u^\top \mathbf{H}_{\psi_t} u.$$

The theoretical minimum is attained when the gradient with respect to u is zero, that is $\nabla_{\psi_t} + \mathbf{H}_{\psi_t} u = 0$ or $u = -\mathbf{H}_{\psi_t}^{-1} \nabla_{\psi_t}$. The Newton's candidate solution for iteration $t + 1$ is often presented as

$$\boldsymbol{\psi}_{t+1} = \boldsymbol{\psi}_t - \gamma \mathbf{H}_{\psi_t}^{-1} \nabla_{\psi_t},$$

where $0 < \gamma \leq 1$ is a modifier of the step size for ensuring convergence (Armijo 1966).

When the method converges the algorithm is quadratically fast, or at least superlinear (Bonnans, Gilbert, Lemarechal, and Sagastizábal 2006): the closer $g(\boldsymbol{\psi})$ is to a quadratic function the better is its Taylor approximation and the better are the algorithm convergence properties.

When the Hessian matrix is almost singular, it is still possible to apply quasi-Newton methods (Luenberger and Ye 2008) to (try to) avoid convergence problems. In our nonlinear regression setting we might have the extra complication of an objective function far from a quadratic shape, so that the (quasi-)Newton method might fail to converge. Although these situations can be thought to be rare, they are often encountered in real applications. For example, in Figure 2 we show a problematic surface that the quasi-Newton BFGS algorithm, as implemented by the base R function `optim()`, is not able to properly explore.

We will try to overcome the issues in the optimization by focusing our search only in a neighborhood of the current estimate, that is using a trust-region around the current solution $\boldsymbol{\psi}_t$. The problem to solve is now

$$\min_{u \in \mathbb{R}^p} g(\boldsymbol{\psi}_t) + \nabla_{\boldsymbol{\psi}_t}^\top u + \frac{1}{2} u^\top \mathbf{H}_{\boldsymbol{\psi}_t} u \qquad \text{s.t. } ||u|| \leq \Delta_t,$$

where $\Delta_t > 0$ is the trust-region radius. Our implementation is based on the exposition of Nocedal and Wright (2006) and follows closely that of Mogensen and Riseth (2018). Briefly, at each iteration we compute the standard Newton's step and accept the new solution if it is within the trust-region. If the Newton's step is outside the admissible region we try an alternative step by a linear combination of the Newton's step and the steepest descent step, with the constraint that its length is exactly equal to the radius Δ_t (dogleg method). This new alternative step is then accepted or rejected on the basis of the actual reduction in the function value. The region radius Δ_{t+1} for iteration $t+1$ is adjusted according to the length and acceptance of the step just computed. For more details, we refer the reader to the extensive discussion found in Nocedal and Wright (2006).

2.5. Algorithm initialization

One of the major challenges in fitting nonlinear regression models is choosing a good starting point for initializing the optimization algorithm. Looking at the example in Figure 2, the choice of $\boldsymbol{\psi}_0 = (1, -1, 1, 0)^\top$ made the BFGS algorithm converge to a local optimum while a global optimum might have been found if a better starting point was instead chosen.

First of all, we present the closed-form maximum likelihood estimates $\hat{\alpha}$ and $\hat{\delta}$ when all other parameters have been fixed. For the logistic function define $h_k = (1 + \nu \exp(-\eta(x_k - \phi)))^{-1/\nu}$. For the log-logistic function define $h_k = (d^\eta / (d^\eta + \nu \phi_+^\eta))^{1/\nu}$. Assume h_k to be known. Our mean function is now

$$\mu_k(\alpha, \delta) = \alpha + \delta h_k,$$

while the residual sum of squares becomes

$$g(\alpha, \delta) = \frac{1}{2} \sum_{k=1}^{m} w_{k \cdot} (\bar{y}_k - \alpha - \delta h_k)^2$$

with gradient

$$\begin{aligned} g^{(\alpha)} &= \alpha \sum_{k=1}^{m} w_k + \delta \sum_{k=1}^{m} w_k h_k - \sum_{k=1}^{m} w_k \bar{y}_k, \\ g^{(\delta)} &= \alpha \sum_{k=1}^{m} w_k h_k + \delta \sum_{k=1}^{m} w_k h_k^2 - \sum_{k=1}^{m} w_k h_k \bar{y}_k. \end{aligned}$$

It is easy to prove that the gradient is equal to zero when

$$\begin{aligned} \hat{\alpha} &= \frac{\left(\sum_k w_k h_k\right) \left(\sum_k w_k h_k \bar{y}_k\right) - \left(\sum_k w_k \bar{y}_k\right) \left(\sum_k w_k h_k^2\right)}{\left(\sum_k w_k h_k\right)^2 - \left(\sum_k w_k\right) \left(\sum_k w_k h_k^2\right)}, \\ \hat{\delta} &= \frac{\left(\sum_k w_k h_k\right) \left(\sum_k w_k \bar{y}_k\right) - \left(\sum_k w_k\right) \left(\sum_k w_k h_k \bar{y}_k\right)}{\left(\sum_k w_k h_k\right)^2 - \left(\sum_k w_k\right) \left(\sum_k w_k h_k^2\right)}. \end{aligned} \tag{6}$$

Our initialization strategy is similar for both the logistic and log-logistic functions and can be summarized in the following steps. Set $\nu = 1$ by default to focus on the 4-parameter version. Define $a = \min\{\bar{\boldsymbol{y}}\} - \epsilon$ and $b = \max\{\bar{\boldsymbol{y}}\} + \epsilon$, where ϵ is a very small number to avoid numerical problems. By construction, $z_k = (\bar{y}_k - a)/(b - a)$ is defined in the range $(0, 1)$. When data is well-behaved and doses span properly the range of the function we can consider a and b to be approximations of $\hat{\alpha}$ and $\hat{\alpha} + \hat{\delta}$ respectively, that is $z_k \approx h_k$. We now have the relationship

$$u_k = \log\left(\frac{z_k}{1 - z_k}\right) \approx -\eta\phi + \eta x_k = \beta_0 + \beta_1 x_k.$$

Remember that $\phi = \log(\phi_+)$ and $x_k = \log(d_k)$, thereby to avoid the logarithm of zero in the log-logistic function we use in our implementation $x_k \approx \log(1 + d_k)$. By fitting the simple linear regression $\boldsymbol{u} = \boldsymbol{\beta}^\top \boldsymbol{x}$ we obtain estimates $\hat{\beta}_0$ and $\hat{\beta}_1$. Our initial estimates are therefore $\eta_0 = |\hat{\beta}_1|$ and $\phi_0 = -\hat{\beta}_0/\hat{\beta}_1$. Note that for a monotonically decreasing function ($\delta < 0$) the coefficient $\hat{\beta}_1$ is negative, therefore to enforce the constraint $\eta > 0$ we use the absolute value of $\hat{\beta}_1$. Finally, we set α_0 and δ_0 at their maximum likelihood estimates by plugging η_0 and ϕ_0 into Equation 6.

In general, parameter $\boldsymbol{\psi}_0 = (\alpha_0, \delta_0, \eta_0, \phi_0, 1)^\top$ is a good starting point for a numerical optimization algorithm. For data that does not behave well we might obtain a very bad starting point because of a poor approximation $z_k \approx h_k$. To avoid problems with corner cases we also perform a grid search over the parameter space of $(\eta, \phi, \nu)^\top$ using a space-filling maximum entropy design with 250 samples (Santner, Williams, and Notz 2018; Dupuy, Helbert, and Franco 2015). For each tested parameter $(\eta, \phi, \nu)^\top$ in the grid we compute the corresponding values of α and δ using Equation 6. The parameter vector corresponding to the smallest residual sum of squares is chosen as our second candidate initial point. To further increase our chances of converging to the global optimum we add to the pool of initial points also two sub-optimal parameters (in the current implementation we use the parameters associated with the fifth and eighth lowest residual sum of squares).

The four chosen parameter vectors are passed in turn as starting points to the `nlminb()` function. `nlminb()` is a very fast and efficient function and for well-behaved data it converges in general to the global optimum. In this case, the best out of the four solutions is also the solution to our optimization problem. To keep refining the solution in cases of problematic data we feed the current solution as a starting point to our own trust region implementation based on analytical gradient and Hessian.

3. Using drda

3.1. General overview

The main function of **drda** is `drda()` with signature

```
drda(formula, data, subset, weights, na.action, mean_function = "logistic4",
  lower_bound = NULL, upper_bound = NULL, start = NULL, max_iter = 1000)
```

The first argument, `formula`, is a symbolic representation in the form `y ~ x` of the model to be fitted, where `y` is the vector of responses and `x` is the vector of log-doses.

`data` is an optional argument, typically a ‘`data.frame`’ object, containing the variables in the model. When `data` is not specified the variables are taken from the environment where the function is being called.

`subset` is a logical vector, or a vector of indices, specifying the portion of `data` to be used for model fitting.

`weights` is an optional argument that specifies the weights to be used for fitting. Usually weights are used in situations where observations are not equally informative, i.e., when it is known that some of the observations should have a smaller or larger impact on the fitting process. If the `weights` argument is not provided then the ordinary least squares method is applied.

`na.action` defines a function for handling `NA`s found in `data`. The default option is to use `na.omit()`, i.e., to remove all data points associated with the missing values.

The `mean_function` argument specifies the model that should be estimated. In the current version of the package the argument can be any of `"logistic5"`, `"logistic4"`, `"logistic2"`, `"gompertz"`, `"loglogistic5"`, `"loglogistic4"`, `"loglogistic2"`, or `"loggompertz"`. Each model is explained in Section 2.1. By default, the 4-parameter logistic function is chosen. Arguments `lower_bound` and `upper_bound` are used for performing constrained optimization. They serve as the minimum and maximum values allowed for the model parameters. They are vectors of length equal to the number of parameters of the model specified by the `mean_function` argument. Values `-Inf` and `Inf` are allowed. The parameters for the 5-parameter (log-)logistic function are listed in the following order: $\alpha, \delta, \eta, \phi, \nu$. For the other models the order is preserved but some of the parameters are excluded. Obviously, values in `upper_bound` must be greater than or equal to the corresponding values in `lower_bound`.

`start` represents an optional vector of starting values for the parameters.

Finally, the `max_iter` argument sets the value for the maximum number of iterations in the optimization algorithm.

After the call to `drda()`, all the common functions expected for a model fit are available: `coef()`, `deviance()`, `logLik()`, `anova()`, `predict()`, `residuals()`, `sigma()`, `summary()`, `vcov()`, `weights()`.

To evaluate the efficacy of the treatment it is also possible to compute the normalized area under or above the curve:

```
nauc(drda_object, xlim, ylim)
naac(drda_object, xlim, ylim)
```

The two-element vector `xlim` defines the interval of integration with respect to `x`. The two-element vector `ylim` defines the theoretical minimum and maximum values for the response variable `y`. Therefore, `xlim` and `ylim` together define a rectangle that is partitioned into two regions by the dose-response curve. The normalized area under the curve (NAUC) is defined as the area of the “lower” rectangle region divided by the total area of the rectangle. The normalized area above the curve (NAAC) is simply its complement, i.e., $1-$ NAUC. Default value of `xlim` is `c(-10, 10)` for the logistic function and `c(0, 1000)` for the log-logistic function. The `xlim` default values were chosen on the basis of dose ranges that are commonly found in the literature. In the majority of real applications the response variable `y` is a relative measure against a control treatment, therefore the default value for `ylim` is chosen to be `c(0, 1)`.

It is also possible to estimate effective doses at particular response levels:

```
effective_dose(drda_object, y, type, level)
```

The `effective_dose()` function estimates the doses at which the fitted curve is equal to the specified response values `y`. Response values can either be passed on a relative scale (`type = "relative"`), in which case the value of `y` is expected to be in the range $(0, 1)$, or on an absolute scale (`type = "absolute"`), in which case the value of `y` can span the whole real line. Additionally, it is possible to specify the confidence interval level via the `level` parameter. Default values are `y = 0.5`, `type = "relative"`, and `level = 0.95`.

3.2. Usage examples

To demonstrate how to use **drda**, we create the following example data:

```
R> dose <- rep(c(0.0001, 0.001, 0.01, 0.1, 1, 10, 100), each = 3)
R> relative_viability <- c(
+    0.877362, 0.812841, 0.883113, 0.873494, 0.845769, 0.999422, 0.888961,
+    0.735539, 0.842040, 0.518041, 0.519261, 0.501252, 0.253209, 0.083937,
+    0.000719, 0.049249, 0.070804, 0.091425, 0.041096, 0.000012, 0.092564)
```

This example imitates an experiment where seven drug doses have been tested three times each. Relative viability measures have been obtained for each dose-replicate pair and, in this case, comprise 21 values in the $(0, 1)$ interval. Note that any finite real number is accepted as a possible valid outcome.

Default fitting

After loading the package, the `drda()` function can be applied directly to the two variables with the `mean_function` selected to be one of the log-logistic functions.

```
R> library("drda")
R> fit_ll4 <- drda(relative_viability ~ dose, mean_function = "loglogistic4")
```

If the user prefers the logistic function then doses have to be log-transformed:

```
R> log_dose <- log(dose)
R> fit_l4 <- drda(relative_viability ~ log_dose)
```

Note that we did not specify the `mean_function` argument because the 4-parameter logistic function is the default model of `drda()`. The following calls are equivalent with respect to the chosen `mean_function`.

```
R> test_data <- data.frame(d = dose, x = log_dose, y = relative_viability)
R> fit_ll4 <- drda(relative_viability ~ dose, mean_function = "loglogistic4")
R> fit_ll4 <- drda(y ~ d, data = test_data, mean_function = "loglogistic4")
R> fit_ll4 <- drda(y ~ d, data = test_data, mean_function = "ll4")
R> fit_l4 <- drda(relative_viability ~ log_dose)
R> fit_l4 <- drda(y ~ x, data = test_data)
R> fit_l4 <- drda(y ~ x, data = test_data, mean_function = "l4")
```

To obtain summaries the user can apply the `summary()` function to the fitted object.

```
R> summary(fit_l4)

Call: drda(formula = y ~ x, data = test_data, mean_function = "l4")

Pearson Residuals:
    Min       1Q   Median       3Q      Max
-1.8163  -0.4575  -0.0234   0.2062   2.0436

Parameters:
                  Estimate Std. Error Lower .95 Upper .95
Maximum             0.8791     0.0260     0.828      0.93
Height             -0.8271     0.0412    -0.908     -0.75
Growth rate         1.1433     0.2754     0.604      1.68
Midpoint at        -2.1177     0.1828    -2.476     -1.76
Residual std err.   0.0654     0.0119     0.042      0.09

Residual standard error on 17 degrees of freedom

Log-likelihood: 29.69
AIC: -49.38
BIC: -44.15

Optimization algorithm converged in 287 iterations
```

The `summary()` function provides information about the Pearson residuals, parameters' and residual standard error estimates, and their 95% confidence intervals. Together with the actual point estimates, the widths of confidence intervals are a good starting point for assessing the reliability of the model fit. The values of the log-likelihood function, AIC, and BIC are also provided. Finally, the `summary()` function warns the user if the algorithm converged and if so, in how many iterations.

Parameter estimates can be accessed using the `coef()` and `sigma()` functions.

```
R> coef(fit_l4)

  alpha   delta     eta     phi
 0.8791 -0.8271  1.1433 -2.1177

R> sigma(fit_l4)

[1] 0.06541
```

It is important to note that the `coef()` function always returns the parameter ϕ on the same scale as the original `x` variable, in this case the log-EC50. Compare the estimated values with those of the log-logistic fit:

```
R> coef(fit_ll4)

  alpha   delta     eta     phi
 0.8791 -0.8271  1.1433  0.1203
```

Our object can be further explored with all the familiar functions expected for a model fit:

```
R> deviance(fit_l4)

[1] 0.07274

R> vcov(fit_l4)

           alpha      delta       eta        phi
alpha  0.0006765 -0.0008055 -0.003198 -0.0014702
delta -0.0008055  0.0016948  0.007015 -0.0005423
eta   -0.0031976  0.0070153  0.075847 -0.0056588
phi   -0.0014702 -0.0005423 -0.005659  0.0334268

R> residuals(fit_l4)

        1         2         3         4         5         6         7
-0.001531 -0.066052  0.004220 -0.002203 -0.029928  0.123725  0.055300
        8         9        10        11        12        13        14
-0.098122  0.008379  0.008889  0.010109 -0.007900  0.133678 -0.035594
       15        16        17        18        19        20        21
-0.118812 -0.008069  0.013486  0.034107 -0.011355 -0.052439  0.040113

R> logLik(fit_l4)

'log Lik.' 29.69 (df=5)

R> predict(fit_l4)

      1       2       3       4       5       6       7       8       9
0.87889 0.87889 0.87889 0.87570 0.87570 0.87570 0.83366 0.83366 0.83366
     10      11      12      13      14      15      16      17      18
0.50915 0.50915 0.50915 0.11953 0.11953 0.11953 0.05732 0.05732 0.05732
     19      20      21
0.05245 0.05245 0.05245

R> predict(fit_l4, newdata = log(c(0.002, 0.2, 2)))

[1] 0.87157 0.34872 0.08404
```

Model comparison and selection

The `anova()` function can be used to compare competing models within the same logistic family of models. The constant model ($\delta = 0$), i.e., a flat horizontal line, is always included by default in the comparisons. When the model being fitted is not the 5-parameter function, the latter is always included as the general reference model in the likelihood-ratio test.

```
R> fit_l2 <- drda(y ~ x, data = test_data, mean_function = "logistic2")
R> anova(fit_l2)

Analysis of Deviance Table

Model 1: a
Model 2: 1 - 1 / (1 + exp(-e * (x - p))) (Fit)
Model 3: a + d / (1 + n * exp(-e * (x - p)))^(1 / n) (Full)

Model 3 is the best model according to the Akaike Information Criterion.

        Resid. Df Resid. Dev Df   AIC   BIC Deviance  LRT Pr(>Chi)
Model 1        20      2.871  1  21.8  23.9
Model 2        19      0.144  2 -39.1 -36.0   -2.728 62.9  2.2e-15 ***
Model 3        16      0.073  5 -47.4 -41.2   -0.071 14.3   0.0025 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the p value refers here to the likelihood-ratio test with a χ^2-distribution asymptotic approximation. In this particular case we are testing the null hypothesis that the complete 5-parameter logistic function is equivalent, likelihood-wise, to our 2-parameter logistic function. The significant result indicates that the 2-parameter logistic function provides a worse fit for the observed data compared to a 5-parameter logistic function.

```
R> fit_gz <- drda(y ~ x, data = test_data, mean_function = "gompertz")
R> fit_l4 <- drda(y ~ x, data = test_data, mean_function = "logistic4")
R> anova(fit_l2, fit_gz, fit_l4)

Analysis of Deviance Table

Model 1: a
Model 2: 1 - 1 / (1 + exp(-e * (x - p)))
Model 3: a + d * exp(-exp(-e * (x - p)))
Model 4: a + d / (1 + exp(-e * (x - p)))
Model 5: a + d / (1 + n * exp(-e * (x - p)))^(1 / n) (Full)

Model 4 is the best model according to the Akaike Information Criterion.

        Resid. Df Resid. Dev Df   AIC   BIC Deviance  LRT Pr(>Chi)
Model 1        20      2.871     21.8  23.9
Model 2         19       0.144  1 -39.1 -36.0   -2.728 62.9  2.2e-15 ***
Model 3         17       0.079  2 -47.7 -42.5   -0.065 12.6   0.0018 **
Model 4         17       0.073  0 -49.4 -44.2   -0.006  1.6
Model 5         16       0.073  1 -47.4 -41.2    0.000  0.0   0.8267
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

These results indicate the 4-parameter logistic function as the best fit for the data. Not only has the model the lowest AIC value, but the likelihood-ratio test (Model 5 vs. Model 4) is also not significant. Indeed, the data was generated from a 4-parameter logistic function with $\boldsymbol{\psi} = (0.86, -0.84, 1, -2)^\top$ and $\sigma = 0.05$. Note that the likelihood-ratio test between the 4-parameter logistic model and the Gompertz model could not be performed because of an equal number of parameters (they are not nested models).

Weighted fitting

In case when not all of the observations should be utilized equally in the model, the weights argument can be provided to the `drda()` function. All the generic functions described above also apply to a weighted fit object.

```
R> weights <- c(
+    0.990868, 1.095238, 0.974544, 0.973318, 1.107001, 1.012844, 1.052806,
+    1.019427, 1.032544, 0.919827, 0.971385, 0.959019, 1.037789, 1.006835,
+    0.969383, 0.935633, 1.016597, 1.011085, 0.982307, 1.066032, 0.959870)
R> fit_wl4 <- drda(y ~ x, data = test_data, weights = weights)
R> summary(fit_wl4)

Call: drda(formula = y ~ x, data = test_data, weights = weights)

Pearson Residuals:
    Min       1Q   Median       3Q      Max
-1.7940  -0.4663  -0.0153   0.2062   2.0384

Parameters:
                  Estimate Std. Error Lower .95 Upper .95
Maximum              0.879     0.0260    0.8278      0.93
Height              -0.827     0.0415   -0.9082     -0.75
Growth rate          1.133     0.2702    0.6029      1.66
Midpoint at         -2.112     0.1901   -2.4843     -1.74
Residual std err.    0.066     0.0121    0.0424      0.09

Residual standard error on 17 degrees of freedom

Log-likelihood: 29.52
AIC: -49.04
BIC: -43.82

Optimization algorithm converged in 290 iterations
```

```
R> weights(fit_wl4)

 [1] 0.9909 1.0952 0.9745 0.9733 1.1070 1.0128 1.0528 1.0194 1.0325 0.9198
[11] 0.9714 0.9590 1.0378 1.0068 0.9694 0.9356 1.0166 1.0111 0.9823 1.0660
[21] 0.9599

R> residuals(fit_wl4, type = "weighted")

        1         2         3         4         5         6         7
-0.001009 -0.068584  0.004677 -0.001524 -0.030796  0.125179  0.058145
        8         9        10        11        12        13        14
-0.097690  0.009904  0.008004  0.009428 -0.008268  0.134622 -0.037250
       15        16        17        18        19        20        21
-0.118485 -0.007782  0.013622  0.034320 -0.010973 -0.053849  0.039578
```

Constrained optimization

The `drda()` function allows the choice of admissible values for the parameters by setting the `lower_bound` and `upper_bound` arguments appropriately. Unconstrained parameters are set to `-Inf` and `Inf` respectively. While setting the constraints manually, one should be careful in choosing the values as the optimization problem might become very difficult to solve within a reasonable number of iterations.

In the next example α is fixed to 1, δ is fixed to -1, the growth rate η is allowed to vary in $[0, 5]$, while the midpoint parameter ϕ is left unconstrained.

```
R> lb <- c(1, -1, 0, -Inf)
R> ub <- c(1, -1, 5,  Inf)
R> fit_cnstr <- drda(y ~ x, data = test_data, lower_bound = lb,
+    upper_bound = ub)
R> summary(fit_cnstr)

Call: drda(formula = y ~ x, data = test_data, lower_bound = lb,
  upper_bound = ub)

Pearson Residuals:
   Min      1Q  Median      3Q     Max
-2.006  -1.046   0.235   0.462   1.002

Parameters:
                  Estimate Std. Error Lower .95 Upper .95
Maximum             1.0000         NA        NA        NA
Height             -1.0000         NA        NA        NA
Growth rate         0.6405     0.1021    0.4404      0.84
Midpoint at        -2.4224     0.2321   -2.8773     -1.97
Residual std err.   0.0869     0.0145    0.0585      0.12
Residual standard error on 19 degrees of freedom

Log-likelihood: 22.55
AIC: -39.1
BIC: -35.97

Optimization algorithm converged in 335 iterations
```

Finally, it is possible to provide an explicit starting point using the `start` argument or change the maximum number of iterations with the `max_iter` argument.

```
R> fit_cnstr <- drda(y ~ x, data = test_data, lower_bound = lb,
+    upper_bound = ub, start = c(1, -1, 0.6, -2), max_iter = 10000)
```

Basic plot functionality

As basic plot functionality, **drda** allows to plot the data, the maximum likelihood curve and the approximate confidence intervals for the curve. Alongside the common `plot()` arguments, it is possible to customize the plot by changing the scale of the x-axis with the argument `base` or the level of the confidence intervals with the `level` argument (defaults to 0.95). The available options for `base` are '`e`', '`2`', and '`10`', with the default setting depending on the scale used for the `x` variable in the model `formula`. The curve midpoint and the corresponding (log-)dose are also highlighted in the plot (see Figure 3).

```
R> fit_l5 <- drda(y ~ x, data = test_data, mean_function = "logistic5")
R> plot(fit_l5)
```

It is possible to plot any number of models within the same figure (see Figure 4).

```
R> plot(fit_l2, fit_l4, fit_gz, base = "10", level = 0.9, xlim = c(-10, 5),
+    ylim = c(-0.1, 1.1), xlab = "Dose", ylab = "Relative viability",
+    cex = 0.9, legend = c("2-p logistic", "4-p logistic", "Gompertz"))
```

Compare against plots of log-logistic functions (see Figure 5):

```
R> fit_ll2 <- drda(y ~ d, data = test_data, mean_function = "loglogistic2")
R> fit_lgz <- drda(y ~ d, data = test_data, mean_function = "loggompertz")
R> plot(fit_ll2, fit_ll4, fit_lgz, base = "10", level = 0.9,
+    xlim = c(0, 100), ylim = c(-0.1, 1.1), xlab = "Dose",
+    ylab = "Relative viability", cex = 0.9,
+    legend = c("2-p log-logistic", "4-p log-logistic", "log-Gompertz"))
```

Treatment efficacy metrics

To obtain a measure of treatment efficacy, functions `nauc()` and `naac()` compute respectively the normalized area under the curve and above the curve. Since our example data refers to viability data, we use here the NAAC measure: the closer the value to 1 the better the treatment effect.

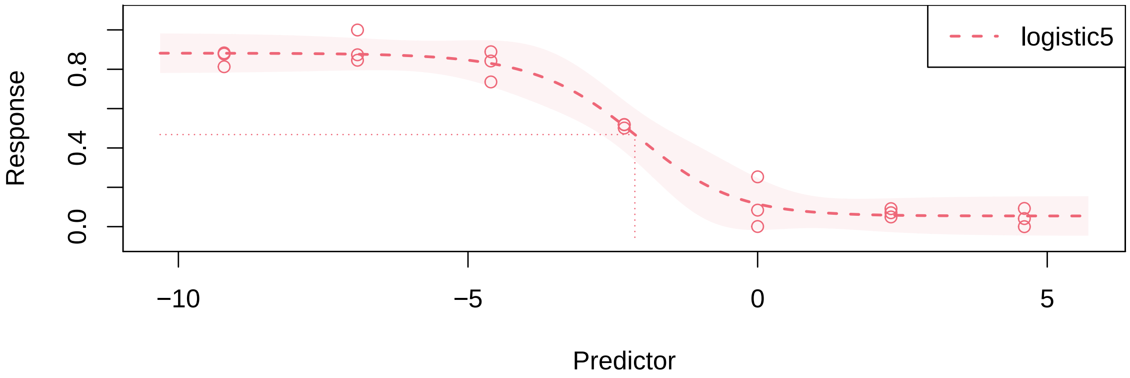

Figure 3: Basic plot of the data and the maximum likelihood curve for the logistic function with 5 parameters.

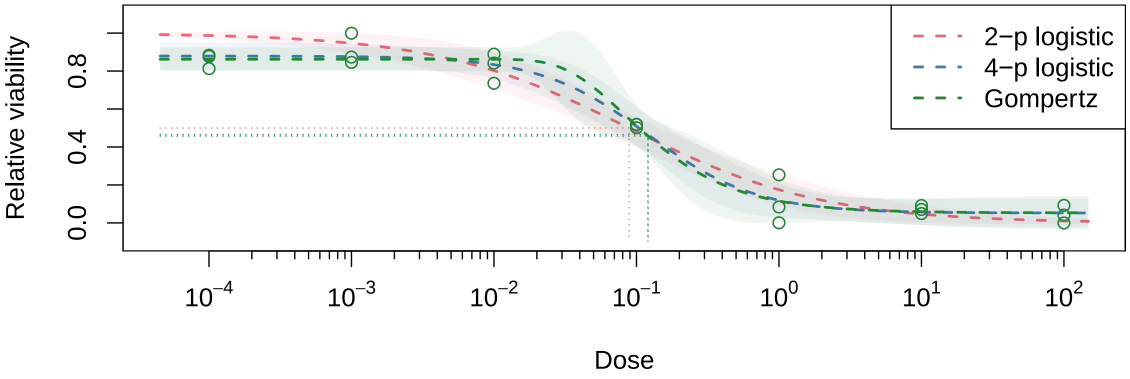

Figure 4: Basic plot of the data and the maximum likelihood curves for several models.

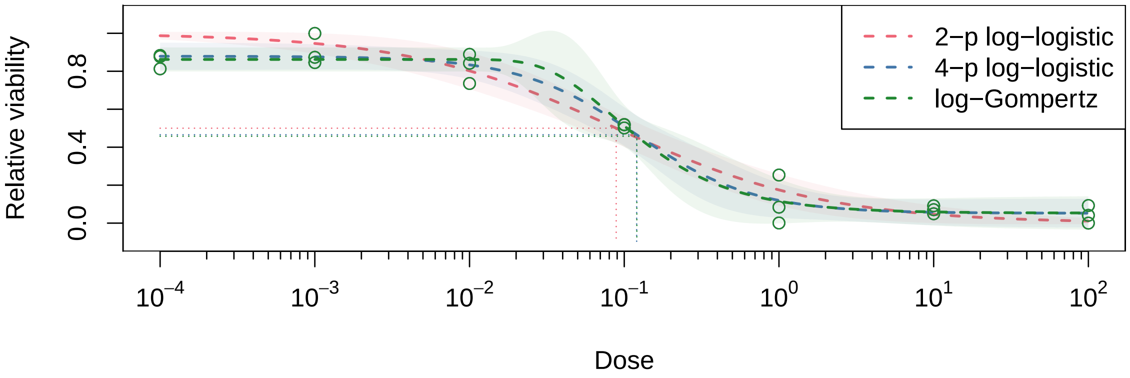

Figure 5: Basic plot of the data and the maximum likelihood curves for several models on the log scale.

```
R> naac(fit_l4)
```

```
[1] 0.622
```

To allow the values to be comparable between different compounds and/or studies, the function sets a hard constraint on both the `x` and `y` variables (see Section 3.1). However, the intervals can be easily changed if needed.

```
R> naac(fit_l4, xlim = c(-2, 2), ylim = c(0.1, 0.9))
```

```
[1] 0.9063
```

Another useful information for treatment efficacy is the effective dose, that is the (log-)dose that produces a specific response.

```
R> effective_dose(fit_l4, y = c(0.75, 0.95))
```

```
     Estimate Lower .95 Upper .95
0.75  -1.1568   -1.6723   -0.6413
0.95   0.4576   -0.6893    1.6045
```

By default `effective_dose()` uses a relative scale for the response, that is the previous results are for the 75% and 95% response. Compare them to the actual values of 0.75 and 0.95:

```
R> effective_dose(fit_l4, y = c(0.75, 0.95), type = "absolute")
```

```
     Estimate Lower .95 Upper .95
0.75   -3.593    -4.234    -2.953
0.95       NA        NA        NA
```

The missing values are a consequence of the fact that the estimated upper bound of the curve is $\hat{\alpha} = 0.88$ and there is no dose such that $\mu(\log(d); \hat{\boldsymbol{\psi}}) = 0.95$.

4. Benchmarking

In this section we evaluate the performance and accuracy of **drda**. We want to compare its performance against the three packages described in Section 1. To do that we select the most widely used model for dose-response curve fitting, that is the 4-parameter log-logistic function. It is the only model implemented in all the four packages we aim to compare. As a control case, we also add to the comparison the `nls()` function from package **stats**. With `nls()` the doses are first log-transformed and the fit is done using the `SSfpl()` self-starting function.

Define, for each parameter, the following possible values:

$$\sigma \in \{0.05, 0.1\}, \quad \alpha \in \{0, 0.2, 0.45\}, \quad \delta \in \{-0.95, 0.3, 1.2\},$$

$$\eta \in \{0.1, 2, 5\}, \quad \phi \in \{0.0001, 1, 100\}.$$

Package	% Convergence	Mean	Minimum	First quartile	Median	Third quartile	Maximum
DoseFinding	–	1.587	0.000	$2.93 \cdot 10^{-10}$	0.012	0.601	60.261
drc	83.8	0.017	0.000	$1.49 \cdot 10^{-5}$	$2.38 \cdot 10^{-4}$	0.004	3.947
drda	100.0	$2.25 \cdot 10^{-7}$	0.000	0.000	0.000	0.000	0.002
nplr	–	1.843	0.000	$1.13 \cdot 10^{-9}$	0.028	1.479	41.705
nls	25.9	$5.65 \cdot 10^{-4}$	0.000	0.000	0.000	0.000	0.189

Table 1: Summary statistics of the RSS relative error for simulated models. A total of 162 parameter vectors were analyzed. For each parameter, 100 datasets with 7 doses and 3 replicates per dose were simulated.

All the possible combinations of the previous values form a set of 162 parameter vectors.

For each parameter vector we simulate 100 datasets, with 7 doses and 3 replicates per dose, from a normal distribution centered on the corresponding log-logistic function. The seven doses are uniformly distributed on the log10 scale: 0.0001, 0.001, 0.01, 0.1, 1, 10, 100. Finally, we apply the fitting function of each package on each one of the 16200 simulated datasets.

For `drm()` from package **drc** we select the 4-parameter log-logistic model with argument `fct = LL.4()` and fix the maximum number of iterations to 10000, similar to `drda()`. For `nplr()` from package **nplr** we set `LPweight` to 0 for the ordinary least squares method. We fix `npars` to four for the 4-parameter log-logistic model. For `fitMod()` from package **DoseFinding** we choose the 4-parameter log-logistic model by setting `model = "sigEmax"` (see Section 2.1). Since the `fitMod()` function requires the user to set constraints on the nonlinear parameters, we use the default values `bnds = defBnds(max(dose))$sigEmax`. For each dataset we store the residual sum of squares (RSS), convergence status, and the elapsed time of running the function.

Since all packages are solving the same optimization problem, i.e., minimization of the residual sum of squares, we define the absolute relative error of package k as

$$\rho_k = \left| 1 - \frac{\mathrm{RSS}_k}{\min\{\mathrm{RSS}_{\mathbf{DoseFinding}}, \mathrm{RSS}_{\mathbf{drc}}, \mathrm{RSS}_{\mathbf{drda}}, \mathrm{RSS}_{\mathbf{nls}}, \mathrm{RSS}_{\mathbf{nplr}}\}} \right| .$$

For practical purposes small absolute relative errors can be considered equivalent to zero.

According to our simulation results (Table 1), **drda** provides the most accurate estimates: in its worst performance **drda** achieved a relative error of just 0.2% from the best solution. Overall, **drda** converged to a solution 100% of the times, **drc** 83.8% of the times, while function `nls()` did so only 25.9% of the times. We cannot provide an accurate convergence rate for **DoseFinding** and **nplr** because the information is missing from their returned fit object.

The higher accuracy comes at a small computational cost as more steps are needed for exploring the parameter space. Our data analysis reveals that `fitMOD()` and `nplr()` are the fastest functions to complete the fit (mean elapsed time of 0.008s and 0.012s respectively). **drda** found the global optimum in 0.514 seconds on average. For completeness, `drm()` and `nls()` needed an average of 0.04 and 0.013 seconds respectively.

We further tested **drda** on a real large-scale drug sensitivity dataset downloaded from the Cancer Therapeutics Response Portal (CTRP, Rees *et al.* 2016; Seashore-Ludlow *et al.* 2015; Basu *et al.* 2013). The data contains cell viability measures for 379533 cell line/drug pairs (887 unique cell lines, 545 unique drugs). The majority of experiments (59.13%) were performed for sixteen drug doses and no replicates, which is only one observation per dose. The relative

Package	% Convergence	Mean	Minimum	First quartile	Median	Third quartile	Maximum
DoseFinding	–	0.445	0.000	0.000	$3.37 \cdot 10^{-4}$	0.018	559.641
drc	97.95	0.08	0.000	$2.40 \cdot 10^{-7}$	$7.55 \cdot 10^{-4}$	0.033	76.804
drda	99.99	$2.76 \cdot 10^{-4}$	0.000	0.000	0.000	0.000	0.432
nplr	–	0.646	0.000	$1.05 \cdot 10^{-10}$	$9.86 \cdot 10^{-9}$	0.007	263.943
nls	36.50	0.006	0.000	0.000	0.000	0.000	1.394

Table 2: Summary statistics of the RSS relative error as measured on the CTRPv2 data.

viability measures span the $(0.0019, 2.864)$ interval. To speed up the benchmark process we sampled at random 15000 datasets from the full set.

Similarly to our simulation study, we fitted the 4-parameter log-logistic model with each one of the four packages. For each cell line-drug-package triple we performed the benchmark and summarized the results in Table 2.

drda is flagged as the absolute best fit in 96.97% of the cases. When we only consider relative errors greater than 1%, the percentage raises to 99.53% (70.87% for **DoseFinding**, 64.95% for **drc**, 76.14% for **nplr**). When compared directly against the other packages, **drda** outperforms **DoseFinding** in 29.07% of the cases (worse for 0.43%), **drc** in 34.81% of cases (worse for 0.02%), and **nplr** in 23.58% of the cases (worse for 0.06%). When `nls()` successfully converged, **drda** was better in 4.9% and worse in 0.02% of the cases.

Similar to the simulation study results, `fitMod()` and `nplr()` are again the fastest functions to complete the fit (mean elapsed time of 0.009s and 0.013s respectively). On average **drda** found the global optimum (or a very close solution) in 0.3 seconds, `drm()` in 0.092 seconds, while `nls()` in 0.019 seconds.

Results shown in this section were obtained with R version 4.2.1, running under Rocky Linux 8.6 with an Intel Xeon Gold 6230 x86_64 Processor, using the Intel oneAPI Math Kernel Library 2022.1 as the BLAS/LAPACK back-end. Because of system-specific numerical instabilities, executing the provided replication materials within a different environment will likely produce minor differences from what is shown here.

5. Summary and discussion

In this paper we introduced the **drda** package, aimed at evaluating dose-response relationship to advance our understanding of biological processes or pharmacological safety. These types of experiments are of high importance in drug discovery, as they establish an essential step for subsequent therapeutic advances. An appropriate interpretation of the experimental data is grounded on a reliable estimation of the dose-response relationship. Therefore, it is imperative to provide advanced optimization methods that allow more accurate estimation of dose-response parameters, and the assessment of their statistical significance.

One of the main limitations of most optimization procedures is their convergence to local solutions. The basic quasi-Newton methods applied to logistic curve fitting are sensitive to the selection of a starting point and to cases where data is non-informative. Our package effectively overcomes the convergence problem as we implement a comprehensive multi-step initialization procedure. In case of problematic data, we additionally rely on our own trust region Newton method implementation based on analytical gradient and Hessian. In addition to standard routines, the package allows a user to evaluate a model fitness via the assessment of confidence intervals for the whole dose-response curve, estimation of effective doses and advanced plot options.

We have compared our package with the three state-of-the-art packages – **DoseFinding**, **drc**, and **nplr**. Using simulations and a large-scale drug screening study, we have shown that **drda** has clearly outperformed the other three packages in terms of accuracy. Despite the fact that our package is on average slower than the other three packages, its gain in accuracy is a favorable compromise. For most, if not all, experimental applications, accuracy has a higher priority.

The current version of **drda** provides optimization tools for continuous data and relies on the family of logistic functions only. In most applications the dose-response relationship is fitted with a logistic function. However, in some specific scenarios, other models can be a better description of the dose-response relationship. For example, a linear no-threshold model might be preferred for fitting dose-response data in radiation protection studies. We plan to extend our package to cover such cases in the future. The package is currently completely implemented in base R, therefore there are still many opportunities for improving its performance, by, for example, refactoring core critical functions in C.

If a researcher is looking for a package providing improved accuracy at a relatively low speed cost, **drda** might provide a viable option.

Package **drda** is available on the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/package=drda. Users are encouraged to contribute to package development at https://github.com/albertopessia/drda.

Acknowledgments

We thank CSC, the Finnish IT center for science, for the computational resources used to perform the simulations.

Research is supported by the European Research Council (ERC) starting grant, No 716063 (DrugComb: Informatics approaches for the rational selection of personalized cancer drug combinations).

References

Abramowitz M, Stegun IA (1965). *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables.* 9th reprint edition. Dover Publications, Mineola.

Akaike H (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control*, **19**(6), 716–723. doi:10.1109/tac.1974.1100705.

Armijo L (1966). "Minimization of Functions Having Lipschitz Continuous First Partial Derivatives." *Pacific Journal of Mathematics*, **16**(1), 1–3. doi:10.2140/pjm.1966.16.1.

Basu A, Bodycombe NE, Cheah JH, Price EV, Liu K, Schaefer GI, Ebright RY, Stewart ML, Ito D, Wang S, Bracha AL, Liefeld T, Wawer M, Gilbert JC, Wilson AJ, Stransky N, Kryukov GV, Dancik V, Barretina J, Garraway LA, Hon CSY, Munoz B, Bittker JA, Stockwell BR, Khabele D, Stern AM, Clemons PA, Shamji AF, Schreiber SL (2013). "An Interactive Resource to Identify Cancer Genetic and Lineage Dependencies Targeted by Small Molecules." *Cell*, **154**(5), 1151–1161. doi:10.1016/j.cell.2013.08.003.

Bonnans JF, Gilbert JC, Lemarechal C, Sagastizábal CA (2006). *Numerical Optimization: Theoretical and Practical Aspects*. 2nd edition. Springer-Verlag, Heidelberg, Germany. doi:10.1007/978-3-540-35447-5.

Bornkamp B, Pinheiro J, Bretz F (2019). ***DoseFinding**: Planning and Analyzing Dose Finding Experiments*. R package version 0.9-17, URL https://CRAN.R-project.org/package=DoseFinding.

Commo F, Bot BM (2016). ***nplr**: N-Parameter Logistic Regression*. R package version 0.1-7, URL https://CRAN.R-project.org/package=nplr.

Dupuy D, Helbert C, Franco J (2015). "**DiceDesign** and **DiceEval**: Two R Packages for Design and Analysis of Computer Experiments." *Journal of Statistical Software*, **65**(11), 1–38. doi:10.18637/jss.v065.i11.

Efron B, Hinkley DV (1978). "Assessing the Accuracy of the Maximum Likelihood Estimator: Observed Versus Expected Fisher Information." *Biometrika*, **65**(3), 457–483. doi:10.1093/biomet/65.3.457.

Fletcher R (2000). *Practical Methods of Optimization*. 2nd edition. John Wiley & Sons, Chichester.

Gsteiger S, Bretz F, Liu W (2011). "Simultaneous Confidence Bands for Nonlinear Regression Models with Application to Population Pharmacokinetic Analyses." *Journal of Biopharmaceutical Statistics*, **21**(4), 708–725. doi:10.1080/10543406.2011.551332.

Liu DC, Nocedal J (1989). "On the Limited Memory BFGS Method for Large Scale Optimization." *Mathematical Programming*, **45**(1), 503–528. doi:10.1007/bf01589116.

Luenberger DG, Ye Y (2008). *Linear and Nonlinear Programming*. International Series in Operations Research & Management Science, 3rd edition. Springer-Verlag, New York. doi:10.1007/978-0-387-74503-9.

Macdougall J (2006). "Analysis of Dose-Response Studies—$\mathrm{E}_{\mathrm{max}}$ Model." In N Ting (ed.), *Dose Finding in Drug Development*, pp. 127–145. Springer-Verlag, New York. doi:10.1007/0-387-33706-7_9.

Mogensen PK, Riseth AN (2018). "**Optim**: A Mathematical Optimization Package for Julia." *Journal of Open Source Software*, **3**(24), 615. doi:10.21105/joss.00615.

Nocedal J, Wright SJ (2006). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering, 2nd edition. Springer-Verlag, New York. doi:10.1007/978-0-387-40065-5.

Oehlert GW (1992). "A Note on the Delta Method." *The American Statistician*, **46**(1), 27–29. doi:10.1080/00031305.1992.10475842.

Pessia A (2022). ***drda**: Dose-Response Data Analysis.* R package version 2.0.2, URL https://CRAN.R-project.org/package=drda.

R Core Team (2022). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rees MG, Seashore-Ludlow B, Cheah JH, Adams DJ, Price EV, Gill S, Javaid S, Coletti ME, Jones VL, Bodycombe NE, Soule CK, Alexander B, Li A, Montgomery P, Kotz JD, Hon CSY, Munoz B, Liefeld T, Dančík V, Haber DA, Clish CB, Bittker JA, Palmer M, Wagner BK, Clemons PA, Shamji AF, Schreiber SL (2016). "Correlating Chemical Sensitivity and Basal Gene Expression Reveals Mechanism of Action." *Nature Chemical Biology*, **12**(2), 109–116. doi:10.1038/nchembio.1986.

Richards FJ (1959). "A Flexible Growth Function for Empirical Use." *Journal of Experimental Botany*, **10**(2), 290–301. doi:10.1093/jxb/10.2.290.

Ritz C, Baty F, Streibig JC, Gerhard D (2015). "Dose-Response Analysis Using R." *PLOS One*, **10**(12), e0146021. doi:10.1371/journal.pone.0146021.

Ritz C, Strebig JC (2016). ***drc**: Analysis of Dose-Response Curves.* R package version 3.0-1, URL https://CRAN.R-project.org/package=drc.

Santner TJ, Williams BJ, Notz WI (2018). *The Design and Analysis of Computer Experiments.* Springer Series in Statistics, 2nd edition. Springer-Verlag, New York. doi:10.1007/978-1-4939-8847-1.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**(2), 461–464. doi:10.1214/aos/1176344136.

Seashore-Ludlow B, Rees MG, Cheah JH, Cokol M, Price EV, Coletti ME, Jones V, Bodycombe NE, Soule CK, Gould J, Alexander B, Li A, Montgomery P, Wawer MJ, Kuru N, Kotz JD, Hon CSY, Munoz B, Liefeld T, Dančík V, Bittker JA, Palmer M, Bradner JE, Shamji AF, Clemons PA, Schreiber SL (2015). "Harnessing Connectivity in a Large-Scale Small-Molecule Sensitivity Dataset." *Cancer Discovery*, **5**(11), 1210. doi:10.1158/2159-8290.cd-15-0235.

Sorensen DC (1982). "Newton's Method with a Model Trust Region Modification." *SIAM Journal on Numerical Analysis*, **19**(2), 409–426. doi:10.1137/0719026.

Steihaug T (1983). "The Conjugate Gradient Method and Trust Regions in Large Scale Optimization." *SIAM Journal on Numerical Analysis*, **20**(3), 626–637. doi:10.1137/0720042.

Affiliation:

Alina Malyutina, Jing Tang, Alberto Pessia
Research Program in Systems Oncology (ONCOSYS)
Faculty of Medicine
University of Helsinki
Haartmaninkatu 8
00290 Helsinki, Finland
E-mail: alina.malyutina@helsinki.fi, jing.tang@helsinki.fi,
academic@albertopessia.com
URL: https://www2.helsinki.fi/en/researchgroups/network-pharmacology-for-precision-medicine/software

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
March 2023, Volume 106, Issue 4 *Submitted:* 2021-06-28
doi:10.18637/jss.v106.i04 *Accepted:* 2022-12-10