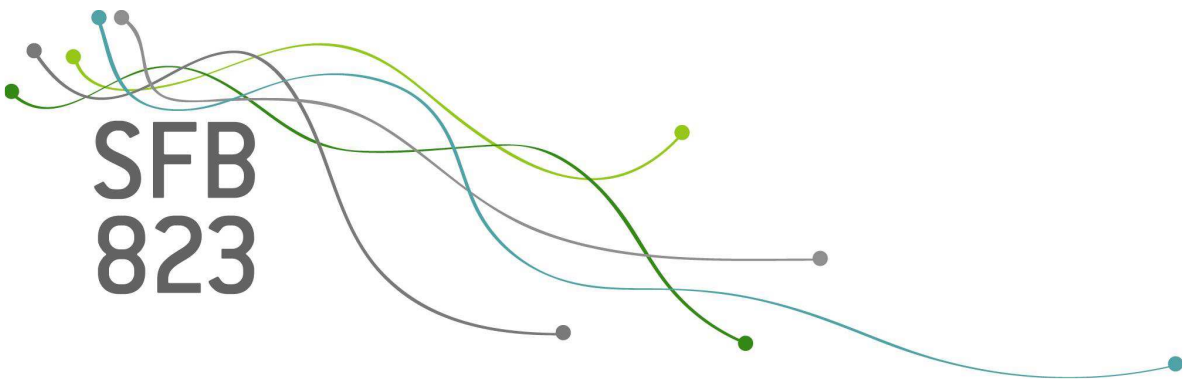
SFB 823

Dose response signal detection by parametric and least squares bootstrap

Patrick Bastian, Holger Dette, Kevin Kokot
Björn Bornkamp, Frank Bretz

Nr. 17/2021

Discussion Paper



Dose response signal detection by parametric and least squares bootstrap

Patrick Bastian, Holger Dette, Kevin Kokot
Fakultät für Mathematik
Ruhr-Universität Bochum
44780 Bochum, Germany

Björn Bornkamp, Frank Bretz
Novartis Pharma AG
4002 Basel, Switzerland

August 9, 2021

1 Introduction

Newly developed pharmaceutical components undergo several trial stages, in particular several dose levels of the compound are compared to placebo in Phase II trials in order to characterize the dose-response relationship. Hsu [1] contains an evaluation of potential benefits of dose-exposure-response modelling by regression models, while Bretz et al [2] offers a review of relevant statistical design and analysis methods. In this report we discuss the problem of dose response signal detection by combining parametric dose response modelling with two resampling strategies.

Modelling the dose-response relation is typically done via a transformation of the function

$$\theta_0 + \theta_1 f(d, \theta^0)\,, \tag{1}$$

where θ_0, θ_1 are linear parameters and f is some (known) non-linear function of the dose variable d that depends in a possibly non-linear fashion on a vector of parameters θ^0. For example (1) could describe the average treatment effect for the dose level d or the expression $\{1 + \exp(-[\theta_0 + \theta_1 f(d, \theta^0)]\}^{-1}$ could represent the probability for the success of a treatment with dose level d. In contrast to the function f the parameters θ_0, θ_1 and θ^0 are typically unknown and estimated from the data.

One major question is of course whether or not there is any dose-response relationship at all, which can be formalized by the equation $\theta_1 = 0$. Naturally the choice of the model f has a large impact on the answer to this question, and usually several candidate models are available for this purpose. This bears a risk of making a wrong decision due to mis-specification

which can not be remedied by naively choosing the model that bests fits the data.

In this report we investigate two bootstrap procedures that aim to rectify these difficulties. Both procedures utilize the extrema of certain functionals to construct the test statistic. More precisely, we first search for optimal parameters to fit each model and then in a second step optimize over the available candidate models. This also facilitates the use of more complex null hypotheses as one only needs to adjust the domain of optimization to adapt the procedure. The first procedure is based on minimizing the least squares error, while the second is based on the log-likelihood ratio.

Section 2 contains a more precise statement of our model assumptionsand also introduces the bootstrap procedures. ´ In Section 3 we present a simulation study to compare the performance of the two procedures with the MCP-Mod method (see Bretz et al. [2] and Pinheiro et al. [3] among others) for different kinds of data (normal, binary, count, survival). Finally, the code of the implementation and some explanation are given in Section 4.

2 Two bootstrap procedures

We consider a response vector $(Y_{1,1}, \dots, Y_{1,n_1}, \dots, Y_{k,1}, \dots, Y_{k,n_k})^\top \in \mathbb{R}^n$, where the first index $i \in \{1, \dots, k\}$ of $Y_{i,j}$ represents the k different dose-levels and the second index $j \in \{1, \dots, n_i\}$ the n_i different observations corresponding to the i-th dose-level. Often the response vector is assumed to be normal distributed in the sense that

$$Y_{i,j} = \theta_0 + \theta_1 f^0(d_i, \theta^0) + \sigma_i \varepsilon_{i,j} \quad i = 1, \dots, k, \;\; j = 1, \dots, n_i \tag{2}$$

where $(\theta_0, \theta_1, {\theta^0}^\top)^\top$ is the vector of parameters of the mean, the errors $\varepsilon_{i,j}$ are independent standard normal distributed random variables and σ_i^2 denotes the variance at dose level d_i $(i = 1, ..., k)$.
We also consider binary, count or survival responses, i.e. binomial, negative binomial or censored exponential/Weibull data, where the expectation is modelled by

$$\ell(\mathbb{E}[Y_{i,j}]) = \theta_0 + \theta_1 f^0(d_i, \theta^0) \tag{3}$$

for some link function ℓ. In the case of a normal distribution we have $\ell(x) = x$, in the case of binomial data one uses

$$\ell(p) = \text{logit}(p) = \log\left(\frac{p}{1-p}\right),$$

where p is the probability of success (which depends on x). In the other cases we have $\ell(x) = \log(x)$.

2.1 Constrained non-linear least squares bootstrap

In this section we consider either the model (2) or (3) and denote by

$$\boldsymbol{\mu} = (\mu_1, \dots, \mu_k)^\top := (\theta_0 + \theta_1 f^0(d_1, \theta^0), ..., \theta_0 + \theta_1 f^0(d_k, \theta^0))^\top$$

the response model vector at the different dose levels $d_1, ..., d_k$. Our interest here is whether or not there actually is any dose-response relationship, i.e. we want to test the following hypothesis:

$$H_0 : \theta_1 = 0 \text{ versus } H_1 : \theta_1 > 0 \tag{4}$$

For this purpose we estimate the parameter $\boldsymbol{\mu}$ by its maximum likelihood estimator $\hat{\boldsymbol{\mu}} = (\hat{\mu}_1, ..., \hat{\mu}_k)^T$. For example, in the case of normal data the i-th component of $\boldsymbol{\mu}$ is given by

$$\hat{\mu}_i = \frac{1}{n_i} \sum_{k=1}^{n_i} Y_{i,k}, \quad i = 1, ..., k$$

In the other cases an iteratively reweighted least squares algorithm(IRLS) as described in the R-package "stats" (see [4] by the R Core Team) is used to obtain the ML-estimator.
Note that in model (2) the errors have a normal distribution, and therefore $\hat{\boldsymbol{\mu}} \sim \mathcal{N}_k(\boldsymbol{\mu}, S)$, where

$$S = \text{diag}\Big(\frac{\sigma_1^2}{n_1}, ..., \frac{\sigma_k^2}{n_k}\Big).$$

In the case of non-normal errors, this relation often holds asymptotically, that is $n_i \to \infty$, such that $n_i/n \to \xi_i > 0$ $(i = 1, \dots, k)$.
We assume that different candidate models are available and denote a typical model by $\mathbf{g}$. The next step consists in fitting different candidate models

$$\mathbf{g}(d, \theta) = \theta_0 + \theta_1 \mathbf{g}^0(d_i, \theta_2) \tag{5}$$

to the estimates $\hat{\boldsymbol{\mu}}$ and $\hat{S}$ using constrained generalized least squares estimation. More precisely we define the vector $\mathbf{g}$ by $\mathbf{g}(d, \theta) = (\mathbf{g}(d_1, \theta), ..., \mathbf{g}(d_k, \theta))^T$ and minimize the criterion

$$\phi(\theta) = (\hat{\boldsymbol{\mu}} - \mathbf{g}(d, \theta))^\top \hat{S}^{-1} (\hat{\boldsymbol{\mu}} - \mathbf{g}(d, \theta)) \tag{6}$$

with respect to the parameter $(\theta_0, \theta_1, \theta_2)$ subject to the restriction $\theta_1 \geq 0$. Here $\hat{S}$ denotes an estimate of the (asymptotic) covariance matrix of $\hat{\boldsymbol{\mu}}$ under $H_0 : \theta_1 = 0$. For this optimization we utilize the nlminb method.

Now, let $\mathbf{g}_1, \dots, \mathbf{g}_K$ denote the different candidate models, then the idea is to take

$$\psi_j = \inf_\theta \phi_j(\theta) \,, \tag{7}$$

where $\phi_j(\theta)$ is the weighted least squares error in (6) calculated for the jth candidate model $\mathbf{g}_j$, that is

$$\phi_j(\theta) = (\hat{\boldsymbol{\mu}} - \mathbf{g}_j(d,\theta))^\top \hat{S}^{-1}(\hat{\boldsymbol{\mu}} - \mathbf{g}_j(d,\theta)) \,. \tag{8}$$

This allows us to take into consideration multiple models at once by taking the minimal error among all competing models and then to compare it to the calculated least squares error under the constant model obtained under the null hypothesis $H_0 : \theta_1 = 0$. More precisely, we look at the difference

$$T = \min_{\theta_0} \psi_0(\theta_0) - \min_{1\le j\le n} \psi_j \tag{9}$$

where we define

$$\psi_0(\theta_0) = (\hat{\boldsymbol{\mu}} - (\theta_0, \dots, \theta_0)^\top)^\top \hat{S}^{-1}(\hat{\boldsymbol{\mu}} - (\theta_0, \dots, \theta_0)^\top). \tag{10}$$

The null hypothesis is then rejected whenever the statistic T is large. The motivation for this decision rule stems from the fact that under the alternative $H_1 : \theta_1 > 0$ a constant model can not minimize the least squares error well, while there are non-constant model(s) that are more capable of doing just that.
Pinheiro et al. [3] also mention that the test statistic T in (9) is a transformation of the likelihood ratio test statistic.

Clearly, we still need to find out what exactly being large means for T, i.e. we need to calculate quantiles of the distribution of the test statistic T under the null hypothesis. For this purpose we use a bootstrap approach and generate m bootstrap-samples as follows:

$$\boldsymbol{\mu}_1^*, \dots, \boldsymbol{\mu}_m^* \sim \mathcal{N}_k((\hat{\theta}_0, \dots, \hat{\theta}_0)^\top, \hat{S}).$$

where

$$\hat{\theta}_0 = \arg\min_{\theta_0} \psi_0(\theta_0),$$

i.e. the best fit under the null hypothesis $H_0 : \theta_1 = 0$.

We now repeat the previous procedure m times starting in the ith step at (5) with the parameters $\boldsymbol{\mu}_i^*$ instead of $\hat{\boldsymbol{\mu}}$. The resulting statistics defined by (9) are denoted $T_1^*, ..., T_m^*$. Subsequently we use the empricial quantiles of this sample for the quantiles of the distribution of T under the null hypothesis. More explicitly we do the following:

For each of the bootstrapped vectors $\boldsymbol{\mu}_i^*$, we minimize the following generalized least squares criterion:

$$\phi^*(\theta) = (\boldsymbol{\mu}_i^* - \mathbf{g}(d,\theta))^\top \hat{S}^{-1}(\boldsymbol{\mu}_i^* - \mathbf{g}(d,\theta)) \tag{11}$$

with respect to the parameter $(\theta_0, \theta_1, \theta^{0\top})^\top$ subject to the restriction $\theta_1 \ge 0$, where the vector $\mathbf{g}$ is defined by $\mathbf{g}(d,\theta) = (\mathbf{g}(d_1,\theta), ..., g(d_k,\theta))^\top$. Next, we define

$$\psi_j^* = \inf_\theta \phi_j^*(\theta) \,,$$

where $\phi_j^*(\theta)$ is the weighted least squares error (11) for the jth candidate model $\mathbf{g}_j$ calculated from the bootstrap data $\boldsymbol{\mu}_i^*$. Using the same test statistic as before we now calculate

$$T_i^* = \min_{\theta_0} \psi_0^*(\theta_0) - \min_{1 \le j \le n} \psi_j^* \tag{12}$$

where we define

$$\psi_0^*(\theta_0) = (\boldsymbol{\mu}_i^* - (\theta_0, \dots, \theta_0)^\top)^\top \hat{S}^{-1} (\boldsymbol{\mu}_i^* - (\theta_0, \dots, \theta_0)^\top).$$

Finally, we calculate the empirical $(1-\alpha)$-quantile $q_{1-\alpha}^{(1)}$ of the sample $T_1^*, ..., T_m^*$ and compare it to the original test statistic T, rejecting the null hypothesis whenever $T \ge q_{1-\alpha}^{(1)}$. This procedure is summarized in Algorithm 1.

Algorithm 1: Least Squares Bootstrap

Result: Test for the hypotheses (4) based on least squares bootstrap

(1) Estimate $\boldsymbol{\mu}$ and S from the data.

(2) Calculate the test statistic

$$T = \min_{\theta_0} \psi_0(\theta_0) - \min_{1 \le j \le n} \psi_j$$

where ψ_j, ϕ_j and ψ_0 are defined in (7), (8) and (10), respectively.

(3) **for** i=*1*, …, m **do**

- Generate bootstrap data

$$\boldsymbol{\mu}_i^* \sim \mathcal{N}_k((\hat{\theta}_0, \dots, \hat{\theta}_0)^\top, \hat{S})$$

where $\hat{\theta}_0 = \arg\min_{\theta_0} \psi_0(\theta_0)$ and $\hat{S}$ is the common estimate of the asymptotic covariance matrix under the null hypothesis.

- Calculate the statistic T_i^* defined in (12)

end

(4) Calculate the empirical $(1-\alpha)$-quantile $q_{1-\alpha}^{(1)}$ of the bootstrap sample $T_1^*, ..., T_m^*$ and reject the null hypothesis in (4), whenever

$$T \ge q_{1-\alpha}^{(1)} \tag{13}$$

2.2 Unconstrained non-linear least squares approach

We proceed as in Section 1.1, where the only difference between the approaches appears when we optimize the least squares expression.

2.3 Parametric bootstrap

In this section we investigate an alternative bootstrap approach to test the hypotheses (4). We consider in each model

$$\mathbf{g}(d, \theta) = \theta_0 + \theta_1 \mathbf{g}^0(, \theta_2) \tag{14}$$

the log-likelihood ratio as objective function

$$\phi = -2 \log \left[\frac{\sup_{\theta, \theta_1 = 0, \beta} \mathcal{L}_g(Y, \theta, \beta)}{\sup_{\theta, \theta_1 \geq 0, \beta} \mathcal{L}_g(Y, \theta, \beta)} \right] \tag{15}$$

where $\mathcal{L}_g(Y, \theta, \beta)$ is the likelihood function corresponding to the model under consideration. Here the vector β contains possible additional nuisance parameters of the distribution of the data. For example $\beta = \sigma^2$ in the case of normal distributed data as considered in model (2). Explicit expressions for the likelihood functionfor the different distributional assumptions can be found below, where we also study two distributional assumption suited for count and survival data. Note that the optimization in the denominator of (15) is restricted to the case $\theta_1 \geq 0$, i.e. the slope in model (14) has to be positive.

Defining ϕ_j to be the log-likelihood ratio (15) for the jth model $\mathbf{g}_j$ under consideration we consider the following statistic:

$$T_{\max} = \max_{1 \leq j \leq K} \phi_j \ .$$

It is expected that this test statistic is large under the alternative $H_1 : \theta_1 > 0$, therefore we propose to reject the null hypothesis $H_0 : \theta_0 = 0$, whenever $T_{\max}$ is large. For the calculation of this statistic one needs estimates for the relevant parameters under the null hypothesis and alternative. As these estimates depend on the distributional assumptions they are presented in separate subsections.

2.3.1 Normal distribution

In the case of normal distributed data we use the maximum likelihood estimators under the null hypothesis $H_0 : \theta_1 = 0$. For the mean we obtain

$$\hat{\theta}_0 = \frac{1}{n} \sum_{l=1}^{k} \sum_{i=1}^{n_k} Y_{l,i} \ , \tag{16}$$

where $n = \sum_{l=1}^{k} n_k$ and for the variance

$$\hat{\sigma_i}^2 = \frac{\sum_{l=1}^{n_i} (Y_{i,l} - \hat{\theta}_0)^2}{n_i}; \quad i = 1, ..., k \tag{17}$$

Next we generate bootstrap data from the model

$$Y^*_{i,j} \sim \mathcal{N}(\hat{\theta}_0, \hat{\sigma}_i^2) \quad 1 \leq i \leq k, 1 \leq j \leq n_i \tag{18}$$

For this bootstrap sample we calculate for each competing model (15) the corresponding likelihood function

$$\mathcal{L}_g(Y^*, \theta, \beta) = -\frac{1}{2}\sum_{i=1}^{n} n_i \log(2\pi\sigma_i^{2,*}) - \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{n_i}\frac{(Y_{i,j}^* - \mathbf{g}(d_i,\theta))^2}{\sigma_i^2} \tag{19}$$

(here $\beta = (\sigma_1^2, \dots, \sigma_k^2)^\top$). Consequently, the log-likelihood ratio is given by

$$\begin{aligned}\phi_{\text{norm}}^* &= -2\log\left[\frac{\sup_{\theta,\theta_1=0,\beta}\mathcal{L}_g(Y^*,\theta,\beta)}{\sup_{\theta,\theta_1\geq 0,\beta}\mathcal{L}_g(Y^*,\theta,\beta)}\right] \\ &= \sum_{i=1}^{k} n_i(\log(2\pi\hat{\sigma}_i^{2,*}) + 1) - \inf_{\theta,\theta_1\geq 0,\sigma}\left[\sum_{i=1}^{k} n_i \log(2\pi\sigma_i^2) + \sum_{i=1}^{k}\sum_{j=1}^{n_i}\frac{(Y_{i,j}^* - \mathbf{g}(d_i,\theta))^2}{\sigma_i^2}\right]\end{aligned} \tag{20}$$

where $\hat{\sigma}_i^{2,*}$ denotes the bootstrap analogue of the variance estimate defined in (17).

Finally, define $\phi_{j,\text{norm}}^*$ as the log-likelihood ratio (20) corresponding to the jth competing model $\mathbf{g}_j$ under consideration and a bootstrap version of the test statistic $T_{\max}$ as follows

$$T_{\max,i}^* = \max_{1\leq j\leq n}\phi_{\text{norm},j,i}^*$$

Here the index j stands for the model under consideration and the index i for the ith bootstrap sample. Repeating this step m-times we can calculate the empirical $(1-\alpha)$-quantile $q_{1-\alpha}$ of the sample $T_{\max,1}^*, ..., T_{\max,m}^*$.

2.3.2 Binomial distribution

In the case of binomial distributed data we consider the model

$$\log\left(\frac{\mathbb{P}(Y_{i,j}=1|d_i)}{1-\mathbb{P}(Y_{i,j}=1|d_i)}\right) = \mathbf{g}(d_i,\theta)$$

and generate data (under the null hypothesis)

$$Y_{i,j}^* \sim \mathcal{B}(\hat{\theta}_0)) \tag{21}$$

(note that $\bar{\theta}_0 = \log\left(\frac{\hat{\theta}_0}{1-\hat{\theta}_0}\right)$ is the maximizer of the log-likelihood function under $H_0 : \theta_1 = 0$).

The next step of the procedure is exactly the same as in the case of normal distributed data, where the log-likelihood is given by

$$\log\left(\mathcal{L}_{\mathbf{g}}(Y^*,\theta,\beta)\right) = \sum_{i=1}^{k}\sum_{j=1}^{n_k} Y_{i,j}^*\mathbf{g}(d_i,\theta) + \sum_{i=1}^{k} n_i \log(1-\text{logit}^{-1}(\mathbf{g}(d_i,\theta))) \tag{22}$$

and

$$\text{logit}^{-1}(x) = \frac{1}{1+\exp^{-x}}$$

is the inverse of the logit function. Consequently the log-likelihood ratio is given by

$$\begin{aligned}\phi^*_{\text{bin}} =& 2 \sup_{\theta,\theta_1>0} \Big[\sum_{i=1}^{k} \sum_{j=1}^{n_k} Y^*_{i,j} \mathbf{g}(d_i,\theta) + \sum_{i=1}^{k} n_i \log(1-\text{logit}^{-1}(\mathbf{g}(d_i,\theta))) \Big] \\ & - 2\Big(\sum_{i=1}^{k} \sum_{j=1}^{n_k} Y^*_{i,j} \hat{\theta}^*_0 - n\log(1-\text{logit}^{-1}(\hat{\theta}^*_0)) \Big)\end{aligned}$$

2.3.3 Count Data

In the case of **count data** we consider the negative binomial model with density

$$p(k|\mu,\phi) = \binom{k+\phi-1}{k} \left(\frac{\mu}{\mu+\phi}\right)^k \left(\frac{\phi}{\mu+\phi}\right)^\phi \quad (k \in \mathbb{N}), \tag{23}$$

such that

$$\log\Big(\mathbb{E}[Y_{i,j}|d_i]\Big) = \mu = \mathbf{g}(d_i,\theta)$$

and generate data (under the null hypothesis)

$$Y^*_{i.j} \sim \mathcal{B}_{neg}(\hat{\theta}_0, \hat{\gamma}) \tag{24}$$

where $\mathcal{B}_{neg}(\mu,\phi)$ denotes the negative Binomial distribution defined by its above probability mass function, $\hat{\theta}_0$ is defined in (16) and $\hat{\gamma}$ is the maximum likelihood estimate for the size parameter under $H_0 : \theta_1 = 0$. Note that we parameterize the negative Binomial distribution in (23) by its mean and size. The remaining step is then the same as in the case of normal distributed data, where the log-likelihood is given by:

$$\begin{aligned}\log(\mathcal{L}_g(Y^*,\theta,\gamma)) = \sum_{i=1}^{k} \sum_{j=1}^{n_k} \Big[& \log(\Gamma(Y^*_{i,j}+\gamma)) - \log(\Gamma(\gamma)) - \log(Y^*_{i,j}!) + Y^*_{i,j}\mathbf{g}(d_i,\theta) + \gamma\log(\gamma) \\ & - (Y^*_{i,j}+\gamma)\log(\exp(\mathbf{g}(d_i,\theta))+\gamma)\Big]\end{aligned} \tag{25}$$

where $\Gamma(x)$ denotes the Gamma function.

2.3.4 Survival Data

In the case of **survival data** we consider the Weibull model with density

$$f(x,\lambda,m) = \frac{m}{\lambda}\Big(\frac{x}{\lambda}\Big)^{m-1} \exp\Big(-x^m/\lambda^m\Big) \quad (x>0) \tag{26}$$

such that

$$\log\left(\mathbb{E}[Y_{i,j}|d_i]\right) = \log(\lambda_i\Gamma(1+1/m)) = \mathbf{g}(d_i,\theta) \tag{27}$$

and generate data (under the null hypothesis)

$$Y^*_{i.j} \sim \min\left(\mathcal{W}\left(\frac{\exp(\hat{\theta}_0)}{\Gamma(1+1/\hat{m})}, \hat{m}\right), 10\right),$$

where $\mathcal{W}(\lambda, m)$ denotes the Weibull distribution with parameters λ and m defined by (26) and $\hat{\theta}_0$ and $\hat{m}$ are the maximum likelihood estimates for $\theta_0 = \log\left(\lambda\Gamma(1+1/m)\right)$ and the shape parameter m under the null hypothesis $H_0 : \theta_1 = 0$.
The remaining step is then the same as in the case of normal distributed data, here the log-likelihood is given by:

$$\begin{aligned}
\log(\mathcal{L}_g(Y^*,\theta,m)) = &\sum_{i,j:Y^*_{i,j}\leq 10}\Bigg[m(\log(\Gamma(1+\frac{1}{m})) - \mathbf{g}(d_i,\theta)) + \log(m) + (m-1)\log(Y^*_{i,j}) \\
&\qquad - \frac{(\Gamma(1+1/m)Y^*_{i,j})^m}{\exp(m\mathbf{g}(d_i,\theta)))}\Bigg] - \sum_{i,j:Y^*_{i,j}>10}\frac{(\Gamma(1+1/m)Y^*_{i,j})^m}{\exp(m\mathbf{g}(d_i,\theta)))} \\
= &\sum_{i,j:Y^*_{i,j}\leq 10}\left[m(\log(\Gamma(1+\frac{1}{m})) - \mathbf{g}(d_i,\theta)) + \log(m) + (m-1)\log(Y^*_{i,j})\right] \\
&- \sum_{i,j}\frac{(\Gamma(1+1/m)Y^*_{i,j})^m}{\exp(m\mathbf{g}(d_i,\theta)))}
\end{aligned} \tag{28}$$

The parametric bootstrap procedure for all four distributional assumptions is summarized in Algorithm 2.

Algorithm 2: Likelihood Ratio Bootstrap

Result: Test for the hypotheses (4) based on parametric bootstrap

(1) Calculate the test statistic

$$T_{\max} = \max_{1\leq j\leq n} \phi_j$$

where

$$\phi_j = -2\log\left[\frac{\sup_{\theta,\theta_1=0,\beta} \mathcal{L}_{g_j}(Y,\theta,\beta)}{\sup_{\theta\geq 0,\beta} \mathcal{L}_{g_j}(Y,\theta,\beta)}\right]$$

and $\mathcal{L}_g(Y,\theta,\beta)$ is defined by (19), (22), (25), (28) in the case of normal, binary, count and survival data respectively.

(2) **for** $i=1,\ldots m$ **do**

- Generate bootstrap data
  - in the case of a normal distribution by (18)
  - in the case of a Binomial distribution by (21)
  - in the case of a count data by (24)
  - in the case of survival data by (27)
- Calculate the bootstrap test statistic

$$T^*_{\max,i} = \max_{1\leq j\leq n} \phi^*_j\ ,$$

where

$$\phi^*_j = -2\log\left[\frac{\sup_{\theta,\theta_1=0,\beta} \mathcal{L}_{g_j}(Y^*,\theta,\beta)}{\sup_{\theta,\theta\geq 0,\beta} \mathcal{L}_{g_j}(Y^*,\theta,\beta)}\right]$$

and $\mathcal{L}_{g_j}(Y^*,\theta,\beta)$ is the likelihood function corresponding to the different data structures (see equations (19), (22), (25) and (28) for normal distributed, Binomial distributed, count and survival data, respectively).

end

(3) Calculate the empirical $(1-\alpha)$ quantile $q^{(2)}_{1-\alpha}$ of the sample $T^*_{\max,1},\ldots,T^*_{\max,m}$ and reject the null hypothesis in (4), whenever

$$T_{\max} \geq q^{(2)}_{1-\alpha}. \tag{29}$$

3 Power Comparison

In this section we provide a comparison of the two bootstrap procedures by means of a simulation study, where we consider the different data types introduced in Section 2. Precise descriptions of the employed scenarios can be found in the respective subsections.

3.1 Normal distributed data

The candidate dose-response models in this section are the linear, emax, exponential and the sigmoid Emax model with $k = 5$ different dose levels $d_i \in \{0, 0.05, 0.2, 0.6, 1\}$ and $n_i = 20$ observations at each dose level. To be precise, the models are defined by

$$g_{lin}(d_i, \theta) = \theta_0 + \theta_1 d_i, \tag{30}$$

$$g_{emax}(d_i, \theta) = \theta_0 + \theta_1 \frac{d_i}{\theta_2 + d_i}, \tag{31}$$

$$g_{semax}(d_i, \theta) = \theta_0 + \theta_1 \frac{d_i^{\theta_3}}{\theta_2^{\theta_3} + d_i^{\theta_3}}, \tag{32}$$

$$g_{exp}(d_i, \theta) = \theta_0 + \theta_1 (\exp(d_i/\theta_2) - 1)\,. \tag{33}$$

We consider two scenarios (A) and (B) for the parameters in these models, which are are taken from Section 3.3 in Gutjahr et al. [5] and listed in Table 1. θ_0 corresponds to the placebo, θ_1 to the slope and θ_2 contains the non-linear parameter(s) of the different models. For illustration purposes we also include in Figure 1 a plot containing the different shapes of the models.
For each scenario we will repeat the Algorithm 1 and 2 1000 times with $m = 100$ bootstrap replications to obtain an empirical approximation of the power and significance levels.

(A)	linear	emax	exponential1	exponential2	sigmoid emax
θ_0	0	0	0	0	0
θ_1	0.4370782	0.5215674	0.01234888	$1.888455 * 10^{-5}$	0.4146834
θ_2	-	0.2	0.2790553	0.1	(0.05, 4)
σ^2	1	1	1	1	1
(B)	linear	emax	exponential1	exponential2	sigmoid emax
θ_0	0	0	0	0	0
θ_1	0.6607332	0.7884558	0.01866786	$2.854786 * 10^{-5}$	0.6268788
θ_2	-	0.2	0.2790553	0.1	(0.05, 4)
σ^2	1	1	1	1	1

Table 1: *Choices for the parameters in the different models under consideration.*

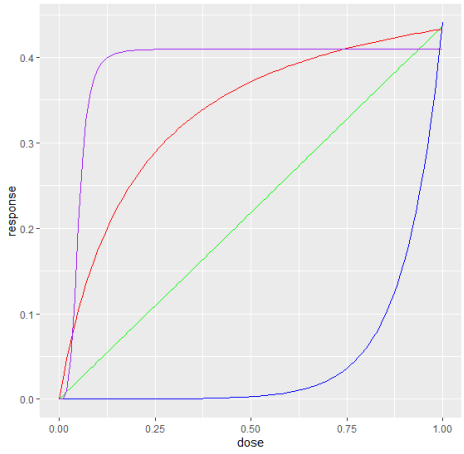


Figure 1: *Shapes of the different models, green corresponds to a linear model, red to an emax one, blue to an exponential one and purple to a sigmoid emax one.*

In Table 2 we display the rejection probabilities of the two tests (13) and (29), where the data was generated under different model assumptions. For the level 5% the results also include the MCP-Mod method developed in Pinheiro et al. [3].
For example, the first line (constant) was generated under the model $g(d, \theta) = \theta_0$ and corresponds to the null hypothesis of no effect in (4). Similarly, the other lines represent several alternatives and in the case of an exponential model we consider two different parameter configurations. The approximation of the nominal level is rather similar for all procedures under consideration, where the parametric bootstrap test (29) slightly exceeds the nominal level. Regarding the power we observe that the MCP-Mod method generally performs a little bit better than the two bootstrap procedures except for the sigmoid emax model. Here both bootstraps achieve significantly higher power. The two bootstrap methods do not differ significantly with respect to the approximation of the nominal level, but the least squares model usually has slightly higher power. For example fixing the nominal power level at 5% we observe that the least squares bootstrap always outperforms the parametric one with only one exception. Finally, we note that MCP-Mod requires knowledge of optimal contrasts depending on unknown parameters of the models which is unrealistic in practice.

level	0.025		0.05			0.1		0.2	
Scenario (A)	lsq	lr	lsq	lr	MCP	lsq	lr	lsq	lr
constant	0.029	0.042	0.061	0.062	0.059	0.107	0.115	0.205	0.227
linear	0.346	0.344	0.4448	0.444	0.478	0.575	0.578	0.736	0.728
emax	0.377	0.344	0.488	0.445	0.447	0.605	0.583	0.753	0.745
exponential1	0.334	0.306	0.430	0.416	0.475	0.560	0.552	0.725	0.691
exponential2	0.283	0.301	0.384	0.411	0.465	0.520	0.532	0.681	0.697
sig	0.373	0.350	0.460	0.448	0.383	0.597	0.572	0.746	0.718
Scenario (B)	lsq	lr	lsq	lr	MCP	lsq	lr	lsq	lr
constant	0.043	0.042	0.065	0.062	0.063	0.103	0.115	0.205	0.227
linear	0.677	0.634	0.768	0.744	0.782	0.851	0.830	0.914	0.922
emax	0.665	0.641	0.753	0.742	0.754	0.848	0.856	0.920	0.932
exponential1	0.641	0.616	0.727	0.696	0.766	0.8833	0.817	0.912	0.902
exponential2	0.610	0.594	0.710	0.693	0.761	0.810	0.802	0.902	0.895
sig	0.644	0.616	0.748	0.716	0.637	0.825	0.810	0.921	0.899

Table 2: *Rejection probabilities of the two bootstrap procedures (13) (lsq) and (29) (lr) for normal distributed data*

In table 3 we study the impact of the sample size on the approximation of the nominal level. We consider the same situation as in Table 2, where now the sample size at each dose level is given by $n_i = 20, 50$ and 100 $(i = 1, \ldots, 5)$. The results demonstrate that the approximation of the nominal level improves with increasing sample size.

level	0.025		0.05		0.1		0.2	
n_i	lsq	lr	lsq	lr	lsq	lr	lsq	lr
20	0.036	0.038	0.066	0.064	0.110	0.113	0.202	0.212
50	0.037	0.028	0.065	0.049	0.109	0.097	0.199	0.189
100	0.036	0.033	0.056	0.052	0.106	0.100	0.194	0.201

Table 3: *Rejection probabilities of the two bootstrap procedures (13) (lsq) and (29) (lr) under the null hypothesis for different sample sizes*

3.2 Binomial, Count and Survival Data

The candidate dose-response models in this section are the emax, exponential and quadratic models with dose levels $d_i \in \{0, 0.05, 0.2, 0.6, 1\}$, more precisely we have

$$g_{quad}(d_i, \theta) = \theta_0 + \theta_1 d_i + \theta_2 d_i^2$$
$$g_{emax}(d_i, \theta) = \theta_0 + \theta_1 \frac{d_i}{\theta_2 + d_i}$$
$$g_{exp}(d_i, \theta) = \theta_0 + \theta_1 (\exp(d_i/\theta_2) - 1)$$

For each scenario we will repeat the Algorithms (1) and (2) 1000 times with $m = 100$ bootstrap replications and a sample size of 20 per dose level. The parameters of the models under which the initial data is generated can be found in Table 4 and are taken from Section 3.2.1 in Pinheiro et al. [3]. In the case of count and survival data we test the hypotheses

$$H_0 : \theta_1 = 0 \quad \text{versus} \quad H_0 : \theta_1 < 0$$

so that the constraints in the Algorithm 1 and 2 are swapped accordingly.

	Binomial			Count			Survival		
	emax	exp	quadratic	emax	exp	quadratic	emax	exp	quadratic
θ_0	-1.734	-1.734	-1.734	2	2	2	0	0	0
θ_1	1.8207	0.01176	4.335	-0.84	-0.005427	-2	-0.7928	-0.005122	-1.8876
θ_2	0.05	0.2	-2.7094	0.05	0.2	1.25	0.05	0.2	1.25

Table 4: *Parameter Choices for binomial, count and survival data*

In Tables 5, 6 and 7 we display the rejection probabilities of the two tests (13) and (29) for binomial, count and survival data respectively. We generated each type of data under several different model assumptions. For the level 5% the results also include the MCP-Mod method developed in Pinheiro et al. [3]. For example, in each table the first line (constant) was generated under the model $g(d, \theta) = \theta_0$ and corresponds to the null hypothesis in (4) of no effect. Similarly, the other lines represent several alternatives. We note again that the MCP-Mod method requires knowledge of the model parameters in order to determine optimal contrasts which seems to be unrealistic in practice.

For binomial data (Table 5) the approximation of the nominal level is rather similar for the bootstrap procedures, where the parametric bootstrap test (29) slightly exceeds the nominal level. The MCP-Mod method on the other hand stays significantly below the nominal level. Regarding the power we observe that the parametric bootstrap and the MCP-Mod method achieve comparable results. On the other hand, the least squares bootstrap is less powerful. For instance, we observe a disparity of more than 10% power when fixing the nominal level at 5% for both emax and quadratic model.

For count data (Table 6) the approximation of the nominal level is rather similar for the bootstrap procedures, with both tests (13) and (29) slightly exceeding the nominal level. Regarding the power we observe that the least squares bootstrap generally slightly outperforms the parametric one. For example for nominal levels 5% and 10% where the parametric bootstrap is outperformed across all models. Both bootstrap methods are in turn outperformed by the MCP-Mod method by a margin of at least 5%.
For survival data (Table 7) both bootstrap procedures exceed the nominal significance level with the least squares bootstrap suffering more severe exceedances. Power-wise we observe similar performance for both bootstrap procedures with the MCP-Mod method matching them

Binomial									
level	0.025		0.05			0.1		0.2	
	lsq	lr	lsq	lr	MCP	lsq	lr	lsq	lr
constant	0.025	0.033	0.052	0.058	0.036	0.099	0.107	0.207	0.198
emax	0.539	0.699	0.670	0.800	0.749	0.819	0.872	0.907	0.940
exponential	0.684	0.742	0.769	0.815	0.857	0.865	0.843	0.930	0.939
quadratic	0.599	0.745	0.704	0.833	0.853	0.822	0.909	0.923	0.948

Table 5: *Rejection probabilities of the two bootstrap procedures (13) (lsq) and (29) (lr) for binomial data*

Count									
level	0.025		0.05			0.1		0.2	
	lsq	lr	lsq	lr	MCP	lsq	lr	lsq	lr
constant	0.034	0.032	0.057	0.057	0.059	0.104	0.110	0.204	0.183
emax	0.683	0.649	0.768	0.736	0.819	0.836	0.831	0.905	0.906
exponential	0.663	0.566	0.760	0.736	0.825	0.843	0.788	0.917	0.908
quadratic	0.626	0.721	0.783	0.778	0.854	0.857	0.855	0.924	0.926

Table 6: *Rejection probabilities of the two bootstrap procedures (13) (lsq) and (29) (lr) for count data*

for emax data, but outperforming them for exponential and quadratic data by 10% and 5% respectively.

Survival									
level	0.025		0.05			0.1		0.2	
	lsq	lr	lsq	lr	MCP	lsq	lr	lsq	lr
constant	0.044	0.030	0.081	0.063	0.036	0.141	0.109	0.246	0.218
emax	0.675	0.683	0.745	0.760	0.749	0.828	0.843	0.896	0.913
exponential	0.688	0.631	0.762	0.732	0.857	0.831	0.823	0.896	0.915
quadratic	0.719	0.714	0.806	0.799	0.853	0.866	0.873	0.932	0.937

Table 7: *Rejection probabilities of the two bootstrap procedures (13) (lsq) and (29) (lr) for survival data*

We conclude this section with a study of the impact of the sample size on the approximation of the nominal level. More precisely, the sample size at each dose level is given by $n_i = 20, 50$ and 100 ($i = 1, \dots, 5$). The corresponding results are displayed in Table 8, 9 and 10. We observe that the approximation of the nominal level generally improves with the sample size. One noteworthy observation is that the approximated level tends increase from $n = 20$ to $n = 50$ while it decreases from $n = 50$ to $n = 100$ (see for example the parametric bootstrap in Table 8 or the least squares bootstrap in Table 9).

Binomial								
level	0.025		0.05		0.1		0.2	
Method	lsq	lr	lsq	lr	lsq	lr	lsq	lr
20	0.025	0.033	0.052	0.058	0.099	0.107	0.0207	0.198
50	0.029	0.037	0.046	0.058	0.107	0.111	0.190	0.215
100	0.026	0.032	0.053	0.053	0.104	0.107	0.190	0.206

Table 8: *Rejection probabilities of the two bootstrap procedures (13) (lsq) and (29) (lr) under the null hypothesis for different sample sizes (binomial data)*

Count								
level	0.025		0.05		0.1		0.2	
Method	lsq	lr	lsq	lr	lsq	lr	lsq	lr
20	0.040	0.032	0.071	0.058	0.125	0.114	0.207	0.187
50	0.040	0.032	0.067	0.061	0.129	0.105	0.219	0.196
100	0.041	0.033	0.064	0.065	0.120	0.104	0.194	0.206

Table 9: *Rejection probabilities of the two bootstrap procedures (13) (lsq) and (29) (lr) under the null hypothesis for different sample sizes for (count data)*

Survival								
level	0.025		0.05		0.1		0.2	
Method	lsq	lr	lsq	lr	lsq	lr	lsq	lr
20	0.059	0.030	0.077	0.063	0.128	0.109	0.223	0.218
50	0.041	0.038	0.070	0.068	0.117	0.121	0.218	0.215
100	0.042	0.032	0.058	0.055	0.113	0.092	0.218	0.182

Table 10: *Rejection probabilities of the two bootstrap procedures (13) (lsq) and (29) (lr) under the null hypothesis for different sample sizes for (survival data)*

Acknowledgements This research was supported by the Collaborative Research Center 'Statistical modeling of nonlinear dynamic processes' (*Sonderforschungsbereich 823, Teilprojekt T1*).

References

[1] Chyi-Hung Hsu. Evaluating potential benefits of dose–exposure–response modeling for dose finding. *Pharmaceutical Statistics*, 8(3):203–215, 2009.

[2] Frank Bretz, Jason Hsu, José Pinheiro, and Yi Liu. Dose finding – a challenge in statistics. *Biometrical Journal*, 50(4):480–504, 2008.

[3] José Pinheiro, Björn Bornkamp, Ekkehard Glimm, and Frank Bretz. Model-based dose finding under model uncertainty using general parametric models. *Statistics in Medicine*, 33(10):1646–1661, 2014.

[4] R Core Team. R: A language and environment for statistical computing. 2013. ISBN 3-900051-07-0.

[5] Georg Gutjahr and Björn Bornkamp. Likelihood ratio tests for a dose-response effect using multiple nonlinear regression models. *Biometrics*, 73(1):197–205, 2017.

4 Appendix A: Code

In this section we include some comments on the code used for the simulations.
Both bootstrap procedures are structured in the same way. First we have a wrapper function in the "ps-calc-..." file which initializes the bootstrap via the "bootstrap" function . This function takes as arguments the sample size, the dose levels, the number of cores used for computation, the data generating models as well as the candidate models among other things. It is the only function that we need to call in order to perform the bootstrap.

The bootstrap function makes use of the code contained in a number of other files, in particular there is the "main..." file which includes the functions needed to perform one bootstrap loop. In both cases there are two methods, one that generates the initial data and calculates the test statistics and a second one that is called upon by the first to deliver the bootstrap quantiles. Both functions make use of the "estimate_..." file that contains the function used to estimate the parameters. (Recall that this amounts to an optimization problem that needs to be solved numerically.)

The "estimate_..." files contain two methods, one for pre-fitting linear parameters (and in the case of normal data for the likelihood ratio bootstrap also the variance) and one that fits the non-linear parameters when using the pre-fitted linear parameters.

There are also the "setModel" and "generate_data" files, the latter contains the functions for generating different kinds of data (initial as well as bootstrap data for the normal, binomial, count and survival settings). The former is used to initialize the different candidate/data generating models as lists containing their names and parameters.

We also include the code, we start with the files specific to the least squares based bootstrap, then the files specific to the likelihood ratio bootstrap and finally we include the files that are used by both (generate_data and setModel).

4.1 Least Squares Bootstrap

4.1.1 ps-calc-GLS

```
library(parallel)
library(gtools)
library(MASS)
library(survival)
source('estimate_GLLS_onesided.R')
source('main_GLS.R')
source('generate_data.R')
source('setModel.R')

Sigma<-1  #standard deviation of original sample
Cores <-3 #amount of cores to be used or computation
Dose<-c(0, 0.05, 0.2, 0.6, 1)  #dose levels
Sample.size<-500 #size of original sample and of bootstrap samples
NSimIn<-100 #amount of inner loops in Bootstrap
Gen.Models<- c('emax') #generating models
Models<-c('emax','exponential','quadratic') #candidate models
Probs <- c(0.025, 0.05, 0.1, 0.2) #specifies levels which are checked



#wrapper function which parallelizes the bootstrapping and
does it for every model in "models" as generating model
bootstrap<-function(nSim, sigma=Sigma, cores=Cores, dose=Dose,
sample.size=Sample.size, nSimIn=NSimIn, models=Models,
 gen.Models=Gen.Models, probs=Probs, setting){
#handing the cores all the relevant information
cl <- makeCluster(cores)
clusterEvalQ(cl, c(library(survival), library(DoseFinding),
 library(gtools), library(parallel), library(matrixStats),
 library(MASS)))
clusterEvalQ(cl, c(source('estimate_GLLS_onesided.R'),
source('main_GLS.R'),
source('generate_data.R'),
source('setModel.R')))
clusterExport(cl, c("cores", "models", "nSimIn", 'sample.size',
"sigma", "dose", "probs", "nSim","setting", 'gen.Models'),
envir=environment())
summary<-matrix(nrow=length(gen.Models)+1,ncol=length(probs))
```

```
for(i in 1:(length(gen.Models)+1)){
if(i==1){
gen.model<-setModel('constant', setting)
}
if(i!=1){
gen.model<-setModel(gen.Models[i-1], setting)
}
clusterExport(cl, c("gen.model"), envir=environment())

rejections<-parSapply(cl, 1:nSim , main, alpha=probs,
nSimIn=nSimIn, sample.size=sample.size, dose=dose, models=models,
true.model=gen.model, setting=setting, sigma=sigma)
for(j in 1:length(probs)){
summary[i,j]<-sum(rejections[j,])
}
}
rownames(summary)<-c('constant', gen.Models)
colnames(summary)<-probs
stopCluster(cl)
return(summary/nSim)
}
```

4.1.2 main-GLS

```
main <- function(dummy = 0, alpha, nSimIn, sample.size, dose, models,
 true.model, setting, sigma){
##############################################################
# Simulates one outer loop of the bootstrap
#
# Args:
#     alpha: vector of significance levels
#     nSimIn: number of inner loops to determine quantiles
#     sample.size: size of original and bootstrap samples
#     dose: vector of different dose levels
#     models: candidate models
#     true.model: model under which original data is generated
#     setting: type of data that is generated
#     sigma: variance in case of normal data
#Returns:
#       vector which contains whether or not nullhypothesis was
#       rejected for the different significance levels
```

```
#
#


#generates doseVector from dose levels and sample.size
n.doses <- length(dose)
doseVec <- c()
for (k in 1:n.doses) doseVec <- c(doseVec, rep(dose[k],
                                    sample.size / n.doses)

#generates original sample
if(setting=='binomial' || setting=='count' || setting=='survival'){
original.sample<-GenOriginalData(samples = 1, model = true.model,
dose.vector = doseVec, setting = setting)
} else{
original.sample<-GenNormData(samples = 1, model = true.model,
dose.vector = doseVec, sd = sigma)
}
bootstrap.parameter <- c()
A<- matrix(c(original.sample, doseVec), nrow=sample.size, ncol=2)
colnames(A)<-c("sample", "dose")
A<-as.data.frame(A)

#fitting data under constant model for different types of data
if(setting=='binomial'){
linear.fit<-glm(sample~as.factor(dose)-1, family = binomial, data=A)
} else if(setting=='count'){
linear.fit<-glm.nb(sample~as.factor(dose)-1,A)
} else if(setting=='survival'){
surv<-Surv(time=c(pmin(original.sample,10)),
event=c(original.sample<10), type='right')
linear.fit<-survreg(surv~as.factor(doseVec)-1, dist='weibull')
} else {
linear.fit<-lm(sample~as.factor(dose)-1,A)
}

 #estimate of the vector that characterizes the response
mu<-as.vector(linear.fit$coefficients)
#estimated covariance matrix corresponding to mu
S<-vcov(linear.fit)
S<-S[-length(dose)-1,-length(dose)-1]
```

```
#contains estimate for mu_0 as well as its GLS-distance \psi_0(mu_0)
fit<-fitConst(mu, S, setting)
mu_0<-rep(fit[1], length(mu))


#taking least squares
for(i in 1:length(models)){
bootstrap.parameter[i]<-estimate(mu, S, models[i], dose, max(dose),
                                 5, setting)[2];
}

#taking minimum of the least square distances
bootstrap.parameter[length(models)+1]<-
                        min(bootstrap.parameter[1:length(models)])

#calculates statistic based on the previously calculated least
#square distances
bootstrap.parameter<-rep(fit[2],length(models)+1)-bootstrap.parameter

#calculates bootstrap statistic quantiles
quantiles<-Quantiles(alpha, nSimIn, sample.size,
                                 dose, models, mu_0, S, setting)

#calculating whether or not H_0 is rejected
rejections<-matrix(nrow=length(models)+1, ncol=length(alpha))
for(i in 1:(length(models)+1)){
for(j in 1:length(alpha)){
if(bootstrap.parameter[i]>=quantiles[i,j]){
rejections[i,j]<-1
}
else{
rejections[i,j]<-0
}
}
}


return(rejections[length(models)+1,])

}
```

```
#####################################################################
# Simulates quantiles of the bootstrap statistics
#
# Args:
#     alpha: vector of significance levels
#     nSimIn: number of inner loops to determine quantiles
#     sample.size: size of original and bootstrap samples
#     dose: vector of different dose levels
#     models: candidate models
#     mu: mean vector for sample generation
#     S: covariance matrix for sample generation
#
# Returns:
#         Matrix which contains bootstrap quantiles(columns) for each
#         model(rows)
#
#

Quantiles<-function(alpha, nSimIn, sample.size, dose, models, mu,
                    S, setting){

#Makes S diagonal
for(i in 1:length(dose)){
for(j in 1:length(dose)){
if(i!=j){
S[i,j]<-0
}
}
}

#initiating store for bootstrap statistics and quantiles
statistics<-matrix(nrow=length(models)+1, ncol=nSimIn)
quantiles<-matrix(nrow=length(models)+1, ncol=length(alpha))
#generate bootstrap data
data <- rmvnorm(nSimIn, mu, S)


for(i in 1:nSimIn){
mu_0<-fitConst(data[i,],S,setting) #fitting mu_0 under constant model
#calculating least squares for different candidate models
```

```
for(j in 1:length(models)){

statistic<-estimate(data[i,], S, models[j], dose, max(dose),5, setting)
statistics[j,i]<-mu_0[2] - statistic[2] #calculating statistic
}
}
#adding the maximum statistic
for(i in 1:nSimIn){
statistics[length(models)+1,i]<-max(statistics[1:length(models),i])
}
#calculating quantiles
for(i in 1:(length(models)+1)){
quantiles[i,]<-quantile(statistics[i,],(1-alpha))
}
return(quantiles)

}

fitConst <- function(mu, S, setting) {
# Fits the constant model to mu and S.
#
# Args:
#   mu: Estimate of the vector which characterizes the distribution of
#       the response vector.
#   S: Estimated covariance matrix of mu.
#
# Returns:
#     Vector containing the following:
#     estimate of the constant,
#     gRSS: The generalized RSS for the estimated constant.

n.doses <- dim(S)[1]
invS <-solve(S)

#estimate the constant
if(setting=='binomial'){
genLS<-function(theta){
as.vector(t(mu - theta) %*% invS %*% (mu - theta))
}
result<-nlminb(1, genLS)
} else{
```

```
genLS<-function(theta){
as.vector(t(mu - theta) %*% invS %*% (mu - theta))
}
result<-nlminb(1, genLS)
}


parameter<- c(result$par, result$objective)
return(parameter)
}
```

4.1.3 estimate_GLLS_onesided

```
library(DoseFinding)
library(parallel)
library(matrixStats)



theta.start.emax<-c(0.2,0.5,1,1.2,1.5)
theta.start.exponential<-c(0.3,0,6,0.9,1,1,2)
theta.start.linlog<-c(0.3,0.6,0.9,1.2,1.5)
theta.start.semax<-matrix(c(0.2,1,0.5,3,1,5,1.2,7,1.5,9),2)
theta.start.quadratic<-c(-3,-2,-1,0,1)
#estimates unknown parameter theta of a model f(x,theta) where x is
#the dose level via constrained least squares optimization
#uses non linear and constrained  least squares minimization
#parameters:
#            mu: response vector
#            S: weight matrix in the squares-equation
#            model: model to be fitted
#            dose: doselevels
#            mD: maximum dose
#            gridpoints: number of points used in the grid for
#                optimization
#returns:
#         estimated values for minimizing slope as well as the minimum
#         of the squares equation in this order
#
#
estimate<-function(mu, S, model, dose, mD, gridpoints, setting){
```

```
if(model=="semax1"| model=="semax2"|model=="semax3"
                                        |model=="semax4")
{
        model<-c("semax")
}

s<-solve(S)
bnds<-defBnds(mD, emax = c(0.001, 1.5)*mD,
exponential = c(0.1, 2)*mD,
logistic = matrix(c(0.001, 0.01, 1.5, 1/2)*mD, 2),
sigEmax = matrix(c(0.001*mD, 0.5, 1.5*mD, 10), 2),
betaMod = matrix(c(0.05,0.05,4,4), 2))
switch(model,

"linear"= {
        genLS<-function(theta){
                slope<-theta[[1]]
                placebo<-theta[[2]]
                lin<-(placebo+dose*slope)

                t(mu-lin) %*% s %*% (mu-lin)
        }
        result<-nlminb(c(1,1), genLS, lower=c(0,-2000))
        parameter<- c(result$par[1], result$objective,
                                result$par[2])
},
"quadratic"= {
        genLS<-function(theta){
                quadr.slope<-theta[[3]]
                placebo<-theta[[2]]
                linear.slope<-theta[[1]]
                quadr<-(placebo+dose*linear.slope+quadr.slope*d

                t(mu-quadr) %*% s %*% (mu-quadr)
        }
        result<-nlminb(c(1,1,1), genLS, upper=c(10,2000,2000))
        parameter<- c(result$par[1], result$objective,
                                        result$par[2])
},
"emax"={
        starting.values<-start.values(mu, s, model, dose, mD,
```

```
                                                    setting)
            genLS<-function(theta){
                    slope<-theta[[1]]
                    placebo<-theta[[2]]
                    nl.par<-theta[[3]]
                    emaxi<-(placebo+slope*dose/(nl.par+dose))

                    t(mu-emaxi) %*% s %*% (mu-emaxi)
            }
            result<-nlminb(c(starting.values,bnds$emax[1]+
                    (bnds$emax[2]-bnds$emax[1])/gridpoints),
                     genLS, lower=c(-100,-1000,bnds$emax[1]),
                     upper=c(100,1000,bnds$emax[2]))
            for(i in 2:gridpoints){
                    cache<-nlminb(c(starting.values,bnds$emax[1]+
                            i*(bnds$emax[2]-bnds$emax[1])/gridpoint
                            ), genLS,
                            lower=c(-100,-1000,bnds$emax[1]),
                            upper=c(100,1000,bnds$emax[2]))
                    if(cache$objective<result$objective){
                            result<-cache
                    }
            }
            parameter<- c(result$par[1], result$objective,
                                    result$par[2])
    },
    "exponential"={
            starting.values<-start.values(mu, s, model, dose, mD,
                                                    setting)
            genLS<-function(theta){
                    slope<-theta[[1]]
                    placebo<-theta[[2]]
                    nl.par<-theta[[3]]
                    expo<-(placebo+slope*(exp(dose/nl.par))-1)

                    t(mu-expo) %*% s %*% (mu-expo)
            }
            result<-nlminb(c(starting.values,bnds$exponential[1]
                    +(bnds$exponential[2]-bnds$exponential[1])
                    /gridpoints), genLS,
                    lower=c(-100,-1000,bnds$exponential[1]),
```

```
                upper=c(100,1000,bnds$exponential[2]))
        for(i in 2:gridpoints){
                cache<-nlminb(c(starting.values,
            bnds$exponential[1]+
                 i*(bnds$exponential[2]-bnds$exponential[1])
                 /gridpoints), genLS,
                 lower=c(-100,-1000,bnds$exponential[1]),
                 upper=c(1000,1000,bnds$exponential[2]))
                if(cache$objective<result$objective){
                        result<-cache
                }
        }
        parameter<- c(result$par[1], result$objective,
                                result$par[2])
},
"linlog"={
        starting.values<-start.values(mu, s, model, dose, mD,
                                setting)
        genLS<-function(theta){
                slope<-theta[[1]]
                placebo<-theta[[2]]
                nl.par<-theta[[3]]
                llog<-(placebo+slope*log(dose+nl.par))

                t(mu-llog) %*% s %*% (mu-llog)
        }
        result<-nlminb(c(starting.values,1), genLS,
                        lower=c(0,-1000,0.0001))
        parameter<- c(result$par[1], result$objective,
                        result$par[2])
},
"semax"={
        #maybe investigate how much the grid approach actually
        #impacts power levels when the method is working
        #properly
        starting.values<-start.values(mu, s, model, dose, mD,
                                        setting)
        llr<-function(theta){
                slope<-theta[[1]]
                placebo<-theta[[2]]
                nl.par<-theta[[3]]
```

```
            nl.par2<-theta[[4]]
            semaxi<-(placebo+slope*dose^(nl.par2)/
                    (nl.par^(nl.par2)+dose^(nl.par2)))

            t(mu-semaxi) %*% s %*% (mu-semaxi)
        }
        #result<-nlminb(c(starting.values, bnds$sigEmax[1,1]
            +(bnds$sigEmax[1,2]-bnds$sigEmax[1,1])/1,
            bnds$sigEmax[2,1]+(bnds$sigEmax[2,2]-
            bnds$sigEmax[2,1])), llr,
            lower=c(0,-1000,bnds$sigEmax[1,1],
                        bnds$sigEmax[2,1]),
            upper=c(1000,1000,bnds$sigEmax[1,2],
                        bnds$sigEmax[2,2]))
        result<-nlminb(c(starting.values,bnds$sigEmax[1,1],
                    bnds$sigEmax[2,1]), llr,
                    lower=c(0,-1000,bnds$sigEmax[1,1],
                                bnds$sigEmax[2,1]),
                    upper=c(10000,1000,bnds$sigEmax[1,2],
                                bnds$sigEmax[2,2]))
        if(gridpoints > 1) {
          for(i in 2:gridpoints){
            cache<-nlminb(c(starting.values,
                bnds$sigEmax[1,1]+
                i*(bnds$sigEmax[1,2]-bnds$sigEmax[1,1])
                /gridpoints, bnds$sigEmax[2,1]
                +i*(bnds$sigEmax[2,2]-bnds$sigEmax[2,1])
                /gridpoints), llr,
                lower=c(0,-1000,bnds$sigEmax[1,1],
                            bnds$sigEmax[2,1]),
                upper=c(10000,1000,bnds$sigEmax[1,2],
                            bnds$sigEmax[2,2]))
            if(cache$objective<result$objective){
                            result<-cache
                    }
            }
        }
        parameter<- c(result$par[1], result$objective,
                            result$par[2])
}
```

```
        )
        return(parameter)
}

start.values<-function(mu, s, model, dose, mD, setting){
        bnds<-defBnds(mD, emax = c(0.001, 1.5)*mD,
        exponential = c(0.1, 2)*mD,
        logistic = matrix(c(0.001, 0.01, 1.5, 1/2)*mD, 2),
        sigEmax = matrix(c(0.001*mD, 0.5, 1.5*mD, 10), 2),
        betaMod = matrix(c(0.05,0.05,4,4), 2))
        switch(model,

        "emax"={
                result<-rep(10000,3)
                for(i in 1:length(theta.start.emax)){
                        genLS<-function(theta){
                                slope<-theta[[1]]
                                placebo<-theta[[2]]
                                nl.par<-theta.start.emax[i]
                                emaxi<-(placebo+slope*dose
                                                        /(nl.par+dose))

                                t(mu-emaxi) %*% s %*% (mu-emaxi)
                        }
                        cache<-nlminb(c(1,1), genLS, upper=c(10,1000))

                        if(cache$objective<result[3]){
                                result<-c(cache$par,cache$objective)
                        }
                }
                parameter<- result[1:2]

        },
        "exponential"={
                result<-rep(10000,3)
                for(i in 1:length(theta.start.exponential)){
                        genLS<-function(theta){
                                slope<-theta[[1]]
                                placebo<-theta[[2]]
                                nl.par<-theta.start.exponential[i]
                                expo<-(placebo+slope*
```

```
                                        (exp(dose/nl.par))-1)

                                t(mu-expo) %*% s %*% (mu-expo)
                        }
                        cache<-nlminb(c(1,1), genLS, upper=c(10,1000))
                        if(cache$objective<result[3]){
                                result<-c(cache$par,cache$objective)
                        }
                }
                parameter<- result[1:2]

        },
        "linlog"={
                result<-rep(10000,3)
                for(i in 1:length(theta.start.linlog)){
                        genLS<-function(theta){
                                slope<-theta[[1]]
                                placebo<-theta[[2]]
                                nl.par<-theta.start.linlog[i]
                                llog<-(placebo+slope*log(dose+nl.par))

                                t(mu-llog) %*% s %*% (mu-llog)
                        }
                        cache<-nlminb(c(1,1), genLS, lower=c(0,-9000))
                        if(cache$objective<result[3]){
                                result<-c(cache$par,cache$objective)
                        }
                }
                parameter<- result[1:2]
        },
        "semax"={
                result<-rep(0,3)
                for(i in 1:(length(theta.start.semax)/2)){
                        llr<-function(theta){
                          slope<-theta[[1]]
                          placebo<-theta[[2]]
                          nl.par<-theta.start.semax[1,i]
                       nl.par2<-theta.start.semax[2,i]
                          semaxi<-(placebo+slope*
                                dose^(nl.par2)/
                                (nl.par^(nl.par2)+dose^(nl.par2)))
```

```
                        t(mu-semaxi) %*% s %*% (mu-semaxi)
                    }
                    cache<-nlminb(c(1,1), llr, lower=c(0,-1000))

                    if((cache$objective)>result[3]){
                        result<-c(cache$par,cache$objective)
                    }
                }
                parameter<- result[1:2]

        }

        )


        return(parameter)

}
```

4.2 Likelihood Ratio Bootstrap

4.2.1 ps-calc-LR

```
library(parallel)
library(gtools)
source('estimate_LR_onesided.R')
source('main_LR_onesided.R')
source('generate_data.R')
source('setModel.R')
Sigma<-1  #standard deviation of original sample
Cores <-3 #amount of cores to be used for computation
Dose<-c(0, 0.05, 0.2, 0.6, 1)   #dose levels
Sample.size<-500  #size of original sample and of bootstrap samples
NSimIn<-100 #amount of inner loops in Bootstrap
genModels<- c('emax') #generating models
Models<-c('emax','exponential','quadratic') #candidate models


Probs <- c(0.025, 0.05, 0.1, 0.2) #specifies levels which are checked
```

```
# wrapper function which parallelizes the bootstrapping and does it
#for every model in "genModels" as generating model
bootstrap<-function(nSim, sigma=Sigma, cores=Cores, dose=Dose,
sample.size=Sample.size, nSimIn=NSimIn, models=genModels,
candidate.Models=Models, probs=Probs, setting){
# handing the cores all the relevant information
cl <- makeCluster(cores)
# set the seed for reproducibility
clusterSetRNGStream(cl, 6545)
clusterEvalQ(cl, c(library(DoseFinding), library(parallel),
library(matrixStats),library(gtools)))
clusterEvalQ(cl, c(source('estimate_LR_onesided.R'),
source('main_LR_onesided.R'),
source('generate_data.R'),
source('setModel.R')))
clusterExport(cl, c("cores", "models", "nSimIn", 'sample.size',
"sigma", "dose", "probs", 'nSim', 'setting'), envir=environment())
summary <- matrix(nrow = length(models)+1,ncol=length(probs))
singleSummaries <- matrix(nrow = (length(models)+1)*
                  length(candidate.Models),ncol=length(probs))
#loop where for each model we calculate the desired rejection rates
for(i in 1:(length(models)+1)) {
if(i == 1) {
gen.model <- setModel('constant', setting)
}
if(i != 1) {
gen.model <- setModel(models[i-1], setting)
}
clusterExport(cl, c("gen.model"), envir = environment())
#bootstrapping
rejections <- parSapply(cl, 1:nSim , Quantiles, alpha=probs,
nSimIn=nSimIn, sample.size=sample.size, dose=dose,
models=candidate.Models, true.model=gen.model, setting=setting)
#bootstrapping with each single candidate model as the sole candidate
for(p in 1:length(candidate.Models)){
singleRejections <- parSapply(cl, 1:nSim , Quantiles, alpha=probs,
nSimIn=nSimIn, sample.size=sample.size, dose=dose,
models=candidate.Models[p], true.model=gen.model, setting=setting)
for(j in 1:length(probs)) {
```

```
singleSummaries[(i-1)*length(candidate.Models)+p,j]<-
                                                sum(singleRejections[j,])

}
}


for(j in 1:length(probs)) {
summary[i,j] <- sum(rejections[j,])
}

}
rownames(summary) <- c('constant', models)
colnames(summary) <- probs

#return(summary/nSim)

#combining single candidate bootstrap and normal bootstrap results
models<-c('constant', models)
counter<-matrix(0L,nrow=((length(models))*(length(candidate.Models)+1))
                                                ncol=length(probs))
rownames<-c()

for(i in 1:length(models)){
rownames<-c(rownames, paste(models[i],':_min'))
for(j in 1:length(candidate.Models)){
rownames<-c(rownames, paste(models[i],':',candidate.Models[j]))
}
}
colnames(counter)<-probs
rownames(counter)<-rownames
for(i in 1:(length(models))){
for(j in 1:length(candidate.Models)){
counter[(i-1)*(length(candidate.Models)+1)+1,]<-summary[i,]
counter[(i-1)*(length(candidate.Models)+1)+1+j,]
                                <-singleSummaries[j+
                                (i-1)*length(candidate.Models),]
}
}
stopCluster(cl)
```

```
return(counter/nSim)

}
```

4.2.2 main-LR-onesided

```
# generates original sample and estimates its parameters as well as
# the bootstrap statistic
# arguments:
#   sample.size: size of the sample
#   dose: dose levels
#   models: different models to be taken into account
#   true.model: model used to generate original sample
# returns:
#   list that contains: 1. fit for theta_0 under H_0
#                       2. the log likelihood ratios for the different
#                          models as well as their maximum
#                       3. for for data specific parameters under H_0
#                          (i.e. variance for normal data)
#
Simulate <- function(sample.size, dose, models, true.model, setting) {

n.doses <- length(dose)
doseVec <- c()
for (k in 1:n.doses) doseVec <- c(doseVec,
                                  rep(dose[k], sample.size / n.doses))
# original sample is being generated
if(!(setting==50||setting==80)){ #count/survival/binomial data
original.sample<-GenOriginalData(samples = 1, model = true.model,
dose.vector = doseVec, setting=setting)
} else { #i.e. normal data
original.sample<-GenNormData(samples = 1, model = true.model,
dose.vector = doseVec, sd = sigma)
}
bootstrap.parameter <- matrix(nrow=length(models)+2, ncol=3)
estimated.llr <- c()



#estimate parameters under the constant model by the llr estimator for
#different types of data
```

```
if(setting=='survival'){ #censor survival data, changes estimator
original.sample<-pmin(original.sample,10)
censored<-sum(original.sample==10)

#estimate theta_0 and the Weibull rate parameter
ll<-function(theta){
placebo<-theta[[1]]
k<-theta[[2]]
n<-length(original.sample)
dummy<-(gamma(1+1/k)^k)/exp(k*placebo)
return( -2*(sum( ( k*(lgamma(1+1/k)-placebo) +log(k)
                  +log(original.sample)*(k-1)-
                  dummy*original.sample^k ) *(original.sample!=10))-
                  sum( ( original.sample==10)*dummy*original.sample^k)))

}
fit<-nlminb(c(1,1),ll,lower=c(-1,0),upper=c(1,100))
ll<-fit$objective
const.fit<-fit$par[1]
sigma.fit<-fit$par[2]

#estimate theta_0
} else if(setting=='binomial') {
const.fit <- mean(original.sample)
sigma.fit<-0

#estimate theta_0 and count data rate parameter
} else if(setting=='count'){
const.fit <- mean(original.sample)
theta_0<-log(const.fit)

ll<-function(theta){
return( -2*(sum(log(gamma(original.sample+theta))-log(gamma(theta))
                  -lgamma(original.sample+1)+original.sample*theta_0+
                   theta*log(theta)-(original.sample+theta)
                  *log(theta+exp(theta_0)))) )
}


fit<-nlminb(1,ll,lower=c(0.000001),upper=c(100))
sigma.fit<-fit$par
```

```
ll<-fit$objective

#normal data, estimate theta_0 and the variance
} else{
const.fit <- mean(original.sample)
sigma.fit<-sum((original.sample-const.fit)^2)/sample.size
}

#calculate test statistic
for(i in 1:length(models)) {

estimated.llr[i] <- estimate(as.vector(original.sample),
                                        diag(1,n.doses), models[i],
                                        doseVec, max(dose), 1, setting,
                                        const.fit, sigma.fit, -ll);
}
estimated.llr[length(models)+1] <- max(estimated.llr)



return(list(const.fit = const.fit, estimated.llr = estimated.llr,
sigma.fit = sigma.fit))
}


# arguments:
#   alpha: vector of desired significance levels
#   nSimIn: number of inner loops, i.e. number of generated bootstrap
#   samples
#   sample.size: sample size for original sample and for the bootstrap
#   dose: vector of different doses
#   models: vector with the different candidate models
#   true.model: model under which data is generated
#
# returns: returns a vector with rejections/acceptances of
#       H_0 (1= rej, 0=acc) for different significance levels
#
Quantiles <- function(dummy=0, alpha, nSimIn, sample.size, dose,
                        models, true.model, setting){
n.doses <- length(dose)
doseVec <- c()
```

```
for (k in 1:n.doses) doseVec <- c(doseVec,
                                  rep(dose[k], sample.size / n.doses))

# calculating test statistic(log-likelihood ratio, llr) and fitting
# under constant model (needed for generating bootstrap data)
sim.list <- Simulate(sample.size, dose, models, true.model, setting)
estimated.llr <- sim.list$estimated.llr


cache <-c();
bootstrap.llr<-matrix(nrow=length(models)+1,ncol=nSimIn);
response<-c()

# loop in which bootstrap data is generated (nSimIn times) and then
# fitted according to the different models to calculate the bootstrap
# test statistic
for(i in 1:nSimIn) {
if(setting=='binomial'){
cache <- GenBinBootData(1, doseVec, sim.list$const.fit);
} else if(setting=='count'){
cache <- GenCountBootData(1, doseVec, sim.list$const.fit,
                          sim.list$sigma.fit);
} else if(setting=='survival'){
cache <- GenSurvBootData(1, doseVec, exp(sim.list$const.fit),
                          sim.list$sigma.fit);
} else {
cache <- GenBootSample(sample.size, sim.list$const.fit,
                          sim.list$sigma.fit)
}


#here the fitting happens
#first we calculate the log likelihood under H_0 as well as some
#parameter estimates
ll<-0
if(setting=='survival'){

#censor survival data
cache<-pmin(cache,10)
censored<-sum(cache==10)
```

```
ll<-function(theta){
placebo<-theta[[1]]
k<-theta[[2]]
n<-length(cache)
dummy<-(gamma(1+1/k)^k)/exp(k*placebo)
return( -2*(sum( ( k*(lgamma(1+1/k)-placebo) +log(k)+log(cache)*(k-1)-
dummy*cache^k ) *(cache!=10) )- sum( ( cache==10)*dummy*(
cache)^k)) )
}
fit<-nlminb(c(1,1),ll,lower=c(-1,0),upper=c(1,100))
ll<-fit$objective
const.fit<-fit$par[1]
sigma.fit<-fit$par[2]


} else if(setting=='binomial') {
const.fit <- mean(cache)
sigma.fit<-0


} else if(setting=='count'){
const.fit <- mean(cache)
theta_0 <- log(const.fit)

ll<-function(theta){
return( -2*(sum(log(gamma(cache+theta))-log(gamma(theta))-
                lgamma(cache+1)+cache*theta_0+theta*log(theta)-
                (cache+theta)*log(theta+exp(theta_0)))) )
}

fit<-nlminb(1,ll,lower=c(0.000001),upper=c(100))
sigma.fit<-fit$par
ll<-fit$objective


} else{ #normal data
const.fit <- mean(cache)
sigma.fit<-sum((cache-const.fit)^2)/sample.size

}
```

```
#calculate likelihood ratios for different models
for(j in 1:length(models)) {
bootstrap.llr[j,i] <- estimate(as.vector(cache), diag(1,n.doses),
                               models[j], doseVec, max(dose), 1,
                               setting, const.fit, sigma.fit, -ll)
}

}

quantiles <- matrix(nrow = length(models)+1, ncol = length(alpha))
# calculating quantiles of the generated bootstrap likelihood ratios
# for the different models
for(i in 1:length(models)) {
quantiles[i,1:length(alpha)]<-quantile(bootstrap.llr[i,],(1-alpha));
}
# putting all the data into the right format
if(length(models) == 1) {
bootstrap.llr[2,] <- bootstrap.llr[1,]
} else {
bootstrap.llr[length(models)+1,]
                        <-colMaxs(bootstrap.llr[1:length(models),])
}
#calculating empirical quantiles of the bootstrap statistic
quantiles[length(models)+1,1:length(alpha)]
                <-quantile(bootstrap.llr[length(models)+1,],(1-alpha))
rejections <- matrix(rep(0,(length(models)+1)*length(alpha)),
                        nrow=length(models)+1, ncol=length(alpha))
#comparing empirical quantiles with llr of original data
#and summing up rejections
for(j in 1:(length(models)+1)){
for(k in 1:length(alpha)){
if(quantiles[j,k] < estimated.llr[j]){rejections[j,k]
                                        <-rejections[j,k]+1}
}
}
return(rejections[length(models)+1,])

}
```

4.2.3 estimate_LR_onesided

```
library(DoseFinding)
library(parallel)
library(matrixStats)

#see documentation of start.values at the bottom

theta.start.emax<-c(0.2,0.5,1,1.2,1.5)
theta.start.exponential<-c(0.3,0.6,0.9,1,1.2)
theta.start.linlog<-c(0.3,0.6,0.9,1.2,1.5)
theta.start.semax<-matrix(c(0.2,1,0.5,3,1,5,1.2,7,1.5,9),2)
theta.start.quadratic<-c(-3,-2,-1,0,1)

#estimates log likelihood ratios for different models and types of data
#
#parameters:
#            mu: response vector
#            S: /
#            model: model to be fitted
#            dose: dose levels
#            mD: maximum dose
#            gridpoints: number of points used in the grid for
#                        optimization
#            setting: type of data(normal, binomial, count, survival)
#            mean: estimate of theta_0
#            sigma: estimate of data specific parameter, e.g.
#                   rate parameter for Weibull survival data
#returns:
#         estimated log likelihood ratio
#
#
estimate<-function(mu, S, model, dose, mD, gridpoints, setting, mean,
                                sigma, l10=0){
if(model=="semax1"| model=="semax2"|model=="semax3"|model=="semax4")
{
model<-c("semax")
}
s<-solve(S)
#bounds for the parameters of the models, taken from the
#DoseFinding package
bnds<-defBnds(mD, emax = c(0.001, 1.5)*mD,
exponential = c(0.1, 2)*mD,
```

```
logistic = matrix(c(0.001, 0.01, 1.5, 1/2)*mD, 2),
sigEmax = matrix(c(0.001*mD, 0.5, 1.5*mD, 10), 2),
betaMod = matrix(c(0.05,0.05,4,4), 2))

#censor data
if(setting=='survival'){
mu<-pmin(mu,10)
}

#define bounds for parameters
if(model=='linear'){
bnd<-matrix(c(0,0,0,0),2)
} else if(model=='emax'){
bnd<-matrix(c(bnds$emax[1],0,bnds$emax[2],0), 2)
} else if(model=='exponential'){
bnd<-matrix(c(bnds$exponential[1],0,bnds$exponential[2],0), 2)
} else if(model=='quadratic'){
bnd<-matrix(c(-10,0,10,0), 2)
} else if(model=='semax'){
bnd<-bnds$sigEmax
}

#calculate likelhood ratios
llr<-function(theta, model, setting){
#define different model structures
slope<-theta[[1]]
placebo<-theta[[2]]
if(model=='linear'){
mod<-placebo+dose*slope

} else if(model=='emax'){
nl.par<-theta[[3]]
mod<-(placebo+slope*dose/(nl.par+dose))

} else if(model=='exponential'){
nl.par<-theta[[3]]
mod<-(placebo+slope*(exp(dose/nl.par))-1)

} else if(model=='quadratic'){
quadr.slope<-theta[[3]]
mod<-(placebo+dose*slope+quadr.slope*dose^2)
```

```
} else if(model=='semax'){
nl.par<-theta[[3]]
nl.par2<-theta[[5]]
mod<-(placebo+slope*dose^(nl.par2)/(nl.par^(nl.par2)+dose^(nl.par2)))

}

#calculate log likelihood ratios for different types of data
if(setting=='binomial'){
theta_0=logit(mean(mu))
-2*(sum(mu*mod+log(1-inv.logit(mod))))+2*sum(mu*theta_0)
+2*length(dose)*log(1-mean(mu))


} else if(setting=='survival'){
k<-theta[[4]]
dummy<-(gamma(1+1/k)^k)/exp(k*mod)
-2*(sum( ( k*(lgamma(1+1/k)-mod) +log(k)+log(mu)*(k-1)-dummy*mu^k)
*(mu!=10) ) -
sum( ( mu==10)*dummy*( mu)^k)) + 110

} else if(setting=='count'){
alpha<-theta[[4]]
n<-length(mu)
-2*(sum(lgamma(mu+alpha)-lgamma(alpha)+alpha*log(alpha)-
(mu+alpha)*log(alpha+exp(mod))-lgamma(mu+1)+mu*mod))+110
} else { #i.e. normal data
AltSigma<-mean((mu-mod)^2)
-2*length(mu)*log(sigma/AltSigma)+2*sum((mod-mu)^2)/AltSigma
}

}
#calculate starting values for optimization
starting.values<-start.values(mu, s, model, dose, mD, setting,
                              sigma, 110)

result<-c()
#optimize likelihood ratios
for(i in 1:gridpoints){
start<-c(starting.values[1:2],
```

```
                        bnd[1,1]+i*(bnd[1,2]-bnd[1,1])/gridpoints, 1,
                        bnd[2,1]+i*(bnd[2,2]-bnd[2,1])/gridpoints)
result[i]<-nlminb(start, llr, model=model, setting=setting,
                  lower=c(-10,-100,bnd[1,1],0.01, bnd[2,1]),
                  upper=c(0,10,bnd[1,2], 10, bnd[2,2]))$objective

}
logl<-min(result)
logl<- -logl


return(logl)
}


#used for prefitting the linear parameters and in the case of
#normal data also for prefitting the variance arguments:
#same as the stimate function
#returns: estimates for theta_0 and theta_1 that will be used as
#starting values in the estimate method above
start.values<-function(mu, s, model, dose, mD, setting, sigma, ll0){


#define bounds for model specific parameters
bnds<-defBnds(mD, emax = c(0.001, 1.5)*mD,
exponential = c(0.1, 2)*mD,
logistic = matrix(c(0.001, 0.01, 1.5, 1/2)*mD, 2),
sigEmax = matrix(c(0.001*mD, 0.5, 1.5*mD, 10), 2),
betaMod = matrix(c(0.05,0.05,4,4), 2))


#censoring data
if(setting=='survival'){
mu<-pmin(mu,10)
}

#define bounds for different model parameters
if(model=='linear'){
bnd<-matrix(c(0,0,0,0),2)
} else if(model=='emax'){
nl.start<-theta.start.emax
```

```
bnd<-matrix(c(bnds$emax[1],0,bnds$emax[2],0), 2)
} else if(model=='exponential'){
nl.start<-theta.start.exponential
bnd<-matrix(c(bnds$exponential[1],0,bnds$exponential[2],0), 2)
} else if(model=='quadratic'){
nl.start<-theta.start.quadratic
bnd<-matrix(c(-10,0,10,0), 2)
} else if(model=='semax'){
nl.start<-theta.start.semax
bnd<-bnds$sigEmax
}
#caluclate likelihood ratios
llr<-function(theta, model, setting, nl.start, i){
slope<-theta[[1]]
placebo<-theta[[2]]
#define different model structures
if(model=='linear'){
mod<-placebo+dose*slope

} else if(model=='emax'){
nl.par<-nl.start[i]
mod<-(placebo+slope*dose/(nl.par+dose))

} else if(model=='exponential'){
nl.par<-nl.start[i]
mod<-(placebo+slope*(exp(dose/nl.par))-1)

} else if(model=='quadratic'){
quadr.slope<-nl.start[i]
mod<-(placebo+dose*slope+quadr.slope*dose^2)

} else if(model=='semax'){
nl.par<-nl.start[1,i]
nl.par2<-nl.start[2,i]
mod<-(placebo+slope*dose^(nl.par2)/(nl.par^(nl.par2)+dose^(nl.par2)))

}

#calculate likelihood ratios for different types of data
if(setting=='binomial'){
theta_0=logit(mean(mu))
```

```
        -2*(sum(mu*mod+log(1-inv.logit(mod))))
        +2*sum(mu*theta_0)+2*length(dose)*log(1-mean(mu))
#note that invlogit(theta_0)=mean(mu)

} else if(setting=='survival'){

k<-sigma
dummy<-(gamma(1+1/k)^k)/exp(k*mod)
-2*(sum(  ( k*(lgamma(1+1/k)-mod) +log(k)+log(mu)*(k-1)-dummy*mu^k)
*(mu!=10)   ) -
sum(    (   mu==10)*dummy*(   mu)^k)) + 110

} else if(setting=='count'){
alpha<-sigma
n<-length(mu)
-2*(sum(lgamma(mu+alpha)-lgamma(alpha)+alpha*log(alpha)-
(mu+alpha)*log(alpha+exp(mod))-lgamma(mu+1)+mu*mod))+110
} else { #i.e. normal data
AltSigma<-mean((mu-mod)^2)
-2*length(mu)*log(sigma/AltSigma)+2*sum((mod-mu)^2)/AltSigma
}

}
result<-c()
par<-c()
#optimize likelihood ratio with different non-linear parameters
for(i in 1:5){
cache<-nlminb(c(1,1), llr, model=model, setting=setting,
                nl.start=nl.start, i=i,
lower=c(0,-10), upper=c(10,10))
result[i]<-cache$objective
if(result[i]==min(result)){
par<-cache$par
}
}

return(par)
```

```
}
```

4.3 Files Used by Both Bootstraps

4.3.1 generate_ data

```
GenNormData <- function(samples, model, dose.vector, sd) {
# Generates data from a specific model.
# 'DoseFinding' package must be loaded.
#
# Args:
#   samples: Amount of samples that should be generated.
#   model: List with to entries 'name' and 'paramters'
#     $name: String which might be one of the following: 'constant',
#            'linear', 'emax', 'exponential', 'linlog'.
#             To do: 'sigEmax', 'quadratic', 'betaMod', 'logistic'.
#     $parameters: The parameters corresponding to the model specified
#                  in 'name'
#   dose.vector: Vector of doses. Length of this vector is equal
#                to the number of observations.
#
# Returns:
#   Matrix of the size 'samples x samplesize'.
#   Each row contains one sample.

samplesize <- length(dose.vector)

# define the mean vector according to the specifications in 'model'
params <- model$parameters
switch(model$name,
constant = mean <- rep(params[1], samplesize),
linear = mean <- linear(dose.vector, params[1], params[2]),
emax = mean<- emax(dose.vector, params[1], params[2], params[3]),
exponential = {
mean <- exponential(dose.vector, params[1], params[2], params[3])
},
exponential1 = {
mean <- exponential(dose.vector, params[1], params[2], params[3])
```

```
},
exponential2 = {
mean <- exponential(dose.vector, params[1], params[2], params[3])
},
linlog = mean <- linlog(dose.vector, params[1], params[2]),
quadratic = mean <- quadratic(dose.vector,
                              params[1], params[2], params[3]),
semax1 = mean <- sigEmax(dose.vector, params[1],
                              params[2], params[3], params[4])
)

data <- sapply(1:samples, function(i)
                             rnorm(samplesize, mean = mean, sd = sd))

return(t(data))
}


GenOriginalData <- function(samples, dose.vector, model, setting) {
# Generates binary data under original model. Can be extended in the
# future.
#
# Args:
#   samples: Amount of samples that should be generated.
#   dose.vector: ??
#   model: ??
#
# Returns:
#   Matrix of the size 'samples x samplesize'.
#   Each row contains one sample.

sample.size <- length(dose.vector)
params <- model$parameters
# calculates model dependent response-vector which will be inv.logit
# transformed into the binomial probability under
# which data is generated
switch(model$name,
constant = value <- c(params[1]),
linear = value <- c(params[1]+params[2]*dose.vector),
emax = value <- c(params[1]
                      +params[2]*dose.vector/(params[3]+dose.vector)),
```

```
exponential = value <-  c(params[1]+
                          params[2]*(exp(dose.vector/params[3])-1)),
exponential1 = value <- c(params[1]+
                          params[2]*(exp(dose.vector/params[3])-1)),
exponential2 = value <- c(params[1]+
                          params[2]*(exp(dose.vector/params[3])-1)),
linlog = value <- c(params[1]+params[2]*log(dose.vector)),
semax1 = value <- c(params[1]+params[2]*dose.vector^(params[4])
                /(params[3]^(params[4])+dose.vector^(params[4]))),
semax2 = value <- c(params[1]+params[2]*dose.vector^(params[4])
                /(params[3]^(params[4])+dose.vector^(params[4]))),
semax3 = value <- c(params[1]+params[2]*dose.vector^(params[4])
                /(params[3]^(params[4])+dose.vector^(params[4]))),
semax4 = value <- c(params[1]+params[2]*dose.vector^(params[4])
                /(params[3]^(params[4])+dose.vector^(params[4]))),
quadratic = value <- quadratic(dose.vector,
                               params[1], params[2], params[3])
)
if(setting=='binomial'){
data <- sapply(1:samples, function(i)
                        rbinom(sample.size, 1, inv.logit(value)))
}
else if(setting=='count'){
data <- sapply(1:samples,
                        function(i) rnbinom(sample.size,
                                    mu=exp(value), size=1))
}
else if(setting=='survival'){
data <- sapply(1:samples,
                        function(i) rexp(sample.size, exp(-value)))
}

return(t(data))
}

#Generates bootstrap data
GenBinBootData <- function(samples, dose.vector, theta) {
sample.size <- length(dose.vector)
data <- sapply(1:samples, function(i) rbinom(sample.size, 1, theta))
return(t(data))
```

```
}
GenCountBootData <- function(samples, dose.vector, mu, size) {
sample.size <- length(dose.vector)
data <- sapply(1:samples, function(i) rnbinom(sample.size,
                                              mu=mu, size=size))
return(t(data))

}
GenSurvBootData <- function(samples, dose.vector, mu, shape) {
sample.size <- length(dose.vector)
data <- sapply(1:samples, function(i) rweibull(sample.size,
                                shape=shape, scale=mu/(gamma(1+1/shape))))
return(t(data))

}

GenBootSample <-function(size, mean, variance){

#Generates bootstrapsample
#
#Args:
#    size: size of generated sample
#    mean: mean of the generated sample
#    variance: variance of the generated sample
#
#Returns:
#    vector with bootstrap sample
sample <-rnorm(size,mean=mean, sd=sqrt(variance));
return(sample)
}
```

4.3.2 setModel

```
# set generating model
setModel<-function(model, setting){
if(setting=='50'){
#setting where optimal test has 50% power in the case of normal data
theta0 <- 0
theta1 <- c(0, 0.4370782) #linear
theta2 <- c(0, 0.5215674, 0.2) #emax
theta3 <- c(0, 1.234888e-02, 0.2790553) #exp1
```

```
theta13<-c(0, 1.888455e-05, 0.1000000) #exp2
theta4 <- c(0, 0.33, 0.2)
theta5 <- c(0, 0.4146834, 0.05, 4) #semax
theta6 <- c(0,  1.050000, 10, 1)
theta7 <- c(0, 1.015625 , 50, 3)
theta8 <- c(0, 1.250000, 100, 2)
theta9 <- c(0, 1.6e-02, -6.4e-05)
theta10 <- c(-1.734, 4.335, -2.7094)
}
if(setting=='80'){
#setting where optimal test has 80% power in the case of normal data
theta0 <- 0
theta1 <- c(0, 0.6607332) #linear
theta2 <- c(0, 0.7884558, 0.2) #emax
theta3 <- c(0, 1.866786e-02, 0.2790553) #exp1
theta13<- c(0, 2.854786e-05, 0.1) #exp2
theta4 <- c(0, 0.33, 0.2) #linlog
theta5 <- c(0, 0.6268788, 0.05, 4) #semax
theta6 <- c(0,  1.050000, 10, 1) #semax
theta7 <- c(0, 1.015625 , 50, 3) #semax
theta8 <- c(0, 1.250000, 100, 2) #semax
theta9 <- c(0, 1.6e-02, -6.4e-05)
theta10 <- c(-1.734, 4.335, -2.7094) #quadratic
}
if(setting=='binomial'){
#setting for binomial test, theta0-theta4 are parameter values
# for original data under constant, linear, emax,
# exponential, linlog models, theta5-8 for different
# sigmoid emax models, theta10 for quadratic model, same holds
# for the other settings
theta0 <- c(0)
theta1 <- c(0, 0.6607332)
theta2 <- c(-1.734, 1.8207, 0.05)
theta3 <- c(-1.734, 0.01176, 0.2)
theta13<-c()
theta4 <- c(0, 0.33, 0.2)
theta5 <- c(0, 1.012500, 2.5, 1)
#theta5 <- c(0, 3, 2.5, 1) # for testing
theta6 <- c(0,  1.050000, 10, 1)
theta7 <- c(0, 1.015625 , 50, 3)
theta8 <- c(0, 1.250000, 100, 2)
```

```
theta9 <- c(0, 1.6e-02, -6.4e-05)
theta10 <- c(-1.734, 4.335, -2.7094)
}
if(setting=='count'){
theta0 <- 0
theta1 <- c(0, 0.6607332) #linear
theta2 <- c(2, -0.84, 0.05) #emax
theta3 <- c(2, -0.005427, 0.2) #exp1
theta13<- c(0, 2.854786e-05, 0.1) #exp2
theta4 <- c(0, 0.33, 0.2) #linlog
theta5 <- c(0, 0.6268788, 0.05, 4) #semax
theta6 <- c(0, 1.050000, 10, 1) #semax
theta7 <- c(0, 1.015625 , 50, 3) #semax
theta8 <- c(0, 1.250000, 100, 2) #semax
theta9 <- c(0, 1.6e-02, -6.4e-05)
theta10<- c(2, -2, 1.25) #quadratic
}
if(setting=='survival') {
theta0 <- 0
theta1 <- c(0, 0.6607332) #linear
theta2 <- c(0,-0.7928,0.05) #emax
theta3 <- c(0,-0.005122,0.2) #exp1
theta13<- c(0, 2.854786e-05, 0.1) #exp2
theta4 <- c(0, 0.33, 0.2) #linlog
theta5 <- c(0, 0.6268788, 0.05, 4) #semax
theta6 <- c(0, 1.050000, 10, 1) #semax
theta7 <- c(0, 1.015625 , 50, 3) #semax
theta8 <- c(0, 1.250000, 100, 2) #semax
theta9 <- c(0, 1.6e-02, -6.4e-05)
theta10 <- c(0, -1.8876, 1.1797) #quadratic
}


gen.model <- c(model)
gen.models <- list(
constant = list(name = 'constant', parameters = theta0),
linear = list(name = 'linear', parameters = theta1),
emax = list(name = 'emax', parameters = theta2),
exponential = list(name = 'exponential', parameters = theta3),
exponential1 = list(name = 'exponential1', parameters = theta3),
exponential2 = list(name = 'exponential2', parameters = theta13),
```

```
linlog = list(name = 'linlog', parameters = theta4),
semax1 = list(name= 'semax1', parameters = theta5),
semax2 = list(name= 'semax2', parameters = theta6),
semax3 = list(name= 'semax3', parameters = theta7),
semax4 = list(name= 'semax4', parameters = theta8),
quadratic = list(name= 'quadratic', parameters = theta10)
)
gen.model<-gen.models[[gen.model]]
return(gen.model)

}
```