Supporting information for: Robust likelihood-based approach for automated optimisation and uncertainty analysis of toxicokinetic-toxicodynamic models

Tjalling Jager*

August 24, 2020

Contents

*DEBtox Research, De Bilt, The Netherlands. Email: tjalling@debtox.nl, http://www.debtox.nl/

1 Introduction

This document provides an extended explanation of the likelihood-based algorithm for optimisation and construction of confidence intervals: here referred to as the parameter-space explorer. This algorithm was developed specifically for the openGUTS project (http://openguts.info/) by DEBtox Research to allow for a fully-automated analysis of survival data. However, the algorithm is more widely applicable for problems with low dimensionality (2-5 free model parameters), and was converted for the BYOM platform under Matlab (see http://debtox.info/byom.html, it was included in the update to version 5.0). This document takes elements from the openGUTS design document (http://openguts.info/download.html), the 'refresher' for the DynModTox course (https://www.debtox.info/dynmodtox.html), and the GUTS e-book [10], adds more explanation, more background, and molds them into a consistent and more didactic treatise. The algorithm applies the 'frequentist' (or better: likelihood-based) viewpoint (as opposed to Bayesian), and constructs confidence intervals by profiling the likelihood function (as opposed to a quadratic approximation).

Section 7 provides a more detailed analysis of the case study from the main text. It uses the openGUTS software, and provides both the stochastic death (SD) and the individual tolerance (IT) analysis. Furthermore, it compares the output to the Bayesian framework as implemented in MORSE (https://cran.r-project.org/package=morse), which also forms the basis for the on-line tool MOSAIC (https://mosaic.univ-lyon1.fr/guts, see [1]).

1.1 Parameter space

A model is a simplification of a part of reality. Every model starts with a set of simplifying assumptions, which are often translated into equations to allow them to be applied. These equations have state variables and model parameters. State variables are the quantities that we are interested in (things that define the state of the system), and that generally change over time. Model parameters are properties of the system that determine how the state variables change over time, and that are generally taken as constants. As an example, consider the one-compartment toxicokinetic (TK) model with first-order kinetics:

$$\frac{dC_i}{dt} = k_u C_w - k_e C_i \tag{1}$$

The state variable is the internal concentration in the organism (C_i), which changes over time. The model parameters are the rate constants k_u and k_e. The external concentration in the environment (C_w) could be a constant, but it could also be time-varying. In any case, it is not a state variable of the system that we describe here (which only considers the organism), and therefore it is referred to as a 'forcing function'; it influences the state variable of the system, but it is not itself influenced by the state variables.

With a certain forcing C_w and a certain initial state $C_i(0)$, the evolution of the internal concentration over time $C_i(t)$ is completely determined by the value of the two (constant) model parameters. Every combination of two values will lead to a different pattern $C_i(t)$

(i.e., a model curve). We can think of all possible values for the model parameters as a 'parameter space'. Since we here have two model parameters, parameter space is two dimensional: each parameter combination is a point on this continuous surface (Fig. 1).

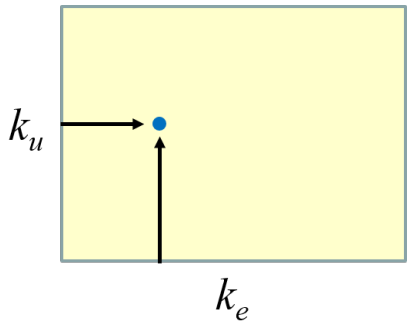


Figure 1: Each combination of two model parameters is a point in a continuous 2-D parameter space (i.e., a plane).

When we have data for internal concentrations over time, we can attempt to find the parameter combination that provides the 'best match' to these data: model fitting or model calibration. This is generally followed by two related questions: how certain are we of these 'best values' (i.e., confidence intervals on model parameters), and how does this uncertainty in model parameters translate to uncertainty in model predictions (i.e., error propagation or confidence intervals on model predictions). Answering all of these questions requires a proper quantification of goodness-of-fit: how can we decide when one model curve is better than another? This requires an additional model: an error model. The error model allows to judge the deviations between model predictions and observations, and is thereby essential to establish the best-fitting parameter values and their confidence intervals (CIs). It is important to realise that the error model is also a model, and that it comes with its own set of assumptions. There are many error models to choose from, and the most appropriate one depends on the nature of the observations; for body residues (continuous or graded) we require a different model than for numbers of survivors (discrete or quantal).

One of the most popular error models is least squares: calculate the differences between each observation and the model prediction at that time point (the residuals), square them, and sum them. Lower values of the sum-of-squares (SSQ) imply a better fit of the model to the data set. The error model thus adds a dimension to parameter space: each parameter combination (each point in the surface of Fig. 1) will now have an associated goodness-of-fit to the data. Our 2-D parameter-space surface thus turns into a 3-D landscape (Fig. 2). In this parameter landscape, the lowest point is our best-fitting combination of model parameters (best-fitting, *given* the data set, and *given* that the error model is 'true').

How can we locate the best-fitting parameter set? In general, we will need to use numerical routines (optimisation methods) to find the lowest point in the landscape. A wide ranges of algorithms exist for this task. If the parameter landscape is well-behaved, as in Figure 2, the task is simple. I call this landscape well-behaved because optimisation routines will have little problems finding the optimum. Imagine placing a ball on the

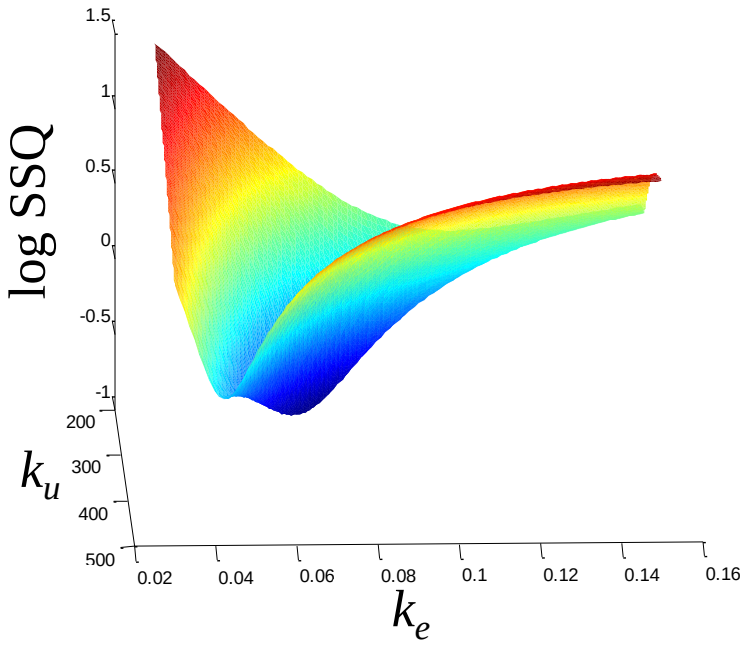


Figure 2: Each combination of two model parameters is a point in a continuous parameter space. Using an error model (here least squares) adds a dimension to parameter space. Colours go from red (poor fit) to blue (good fit).

landscape. No matter where we place the ball, it will always roll to the overall lowest point in the landscape. However, the landscape may not always be well behaved. The actual shape depends on the model for the system, the model for the error, and on the data set. In almost all cases, we do not know *a priori* whether our optimisation problem will be well-behaved or not. The landscape may contain multiple dips (local minima), such that the ball may roll to a place that is not the overall lowest point in the landscape (the global minimum). There may not even be a real dip: the landscape may slope gently down in one direction and never reach a lowest point, such that ball will continue to roll on to ever-higher parameter values. This latter case constitutes an identifiability problem: the data do not contain sufficient information to constrain one or more of the model parameters. These issues of local minima and identifiability are serious problems for model optimisation, and this problem becomes worse when there are more free parameters. For two parameters, we need to navigate a 3-D landscape, but for three parameters we'll have to deal with a 4-D landscape, etcetera.

Apart from the best-fit parameter combination, we are usually also interested in confidence intervals (CIs) on these parameters: how certain can we be about the best value of the model parameters based on the available data? It is easy to see that the CIs relate to the shape of the parameter landscape around the best value: parameter combinations that provide a fit that is not too much worse from the best fit will be part of a CI. In other words, we are interested in the parameter landscape up to a certain height above the best value. In the 2-D plane of the model parameters, we could plot 'lines of equal SSQ' that

would inform about the joint uncertainty in the parameters (illustrated in Fig. 3). In this example, the lines are elliptical, which implies a correlation between the two parameters. The joint CI of the model parameters will be one such ellipse: all parameter combinations within that ellipse will yield a fit that is not too much worse from the best fit. The CIs of the single parameters will be related to the edges of one such an ellipse (explained in the next chapter).

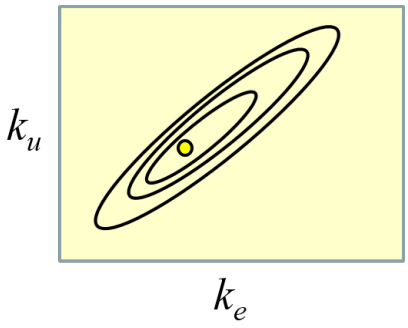


Figure 3: Lines of 'equal SSQ' plotted around the best-fit parameter combination (yellow point).

The challenge is thus to map the parameter landscape efficiently, such that we know the global best parameter combination and the shape of the landscape surrounding it. In principle, we could apply brute force and calculate the goodness-of-fit for a large amount of points in parameter space to create maps as in Figures 2 and 3. In the two-parameter case, this is doable. If we take 100 regular points for each parameter, we can construct a nice smooth landscape with $100 \times 100 = 10^4$ points (i.e., parameter combinations for which we need to calculate the SSQ). However, in higher dimensions (i.e., more parameters to fit), it becomes increasingly difficult to map this landscape efficiently. With four free parameters (as in a typical GUTS fit), we would already need to calculate 10^8 SSQs (this relates to the 'curse of dimensionality'). And, there will be no easy way to display the five dimensions of our landscape in one simple plot. To make matters worse, taking 100 points for each parameter will generally not be sufficient, unless we have a clear *a priori* idea of where the interesting part of parameter space is. In our toxicokinetics model of Eq. 1, the relevant range of values for k_e and k_u may span several orders of magnitude, rapidly requiring more detail than 100 points can generate.

In general, we should therefore forget about mapping and visualising the entire parameter landscape in detail. In higher dimensions, mapping can only be efficient if we can narrow down the search range to where the good fits are located. In general, there will be large parts of parameter space that contain nonsensical fits and are thereby uninteresting. We thus could use more efficient methods to locate and zoom in on that part of parameter space where the most reasonable fits are located. Unfortunately, the focus of almost all existing optimisation algorithms is on finding the best-fitting parameter set, and not on mapping out the parameter landscape around it, for the CIs. Since we generally would like to have CIs, and since robust optimisation requires knowledge of the shape of the

landscape anyway, it would be more efficient to combine both tasks. We will see later that this knowledge about the parameter landscape is essential to quantify uncertainty about model predictions as well.

As an aside, Bayesians *do* tend to map the parameter landscape around the best fit, which in their framework represents the posterior distribution of the model parameters. They often use Markov-Chain Monte Carlo (MCMC) sampling for this purpose to create a sample from the posterior distribution, which can be plotted similarly to Figure 2, but then as a joint probability distribution: the other way around, as higher values of the posterior probability imply a better fit. From that sample, the best-fitting parameter set can be found, and CIs on the model parameters can be generated. Furthermore, the sample can be used for error propagation to model predictions (this is rather straightforward for Bayesians due to the fact that they treat the parameter landscape as a probability distribution). MCMC methods can also be applied for frequentist applications to map the parameter landscape, but they have a few disadvantages. Firstly, they need a starting point in the landscape; an incorrect starting point can leave the algorithm stuck in a local minimum. Secondly, the aim of MCMC is not to explicitly map the landscape, but to provide a random sample from a probability distribution. Therefore, the final sample will contain many points close to the best value and only few in the tails. This is illustrated in Figure 4. That is fine for Bayesians, but not so nice for frequentists. Frequentists need to know the actual height at each sampling point (something that MCMC routines do not automatically provide), and they need detail on the edges to calculate the joint-confidence ellipses. Finally, MCMC algorithms do not react favourably to parameters running away to infinity, or parameters that are strictly bounded (e.g., to values above zero or below 1), and it is not trivial to check that the algorithm has yielded a proper representation of the landscape (see also Section 7).

In summary, an efficient approach to frequentist optimisation requires mapping out the parameter landscape in the relevant part of parameter space (where the best fits are located). This map is then used to 1) find the best-fitting parameter combination, 2) to construct CIs on the model parameters, and 3) to help error propagation to model predictions. For the openGUTS project, an additional requirement for the mapping algorithm was that it should work without input from the user (other than the data set). Most users of openGUTS will not be experts in statistics and numerical optimisation, but this should not preclude a robust statistical inference for any data set. However, it is impossible to guarantee that an algorithm will work in all conceivable cases; the user will at least need the expertise to identify potential problems.

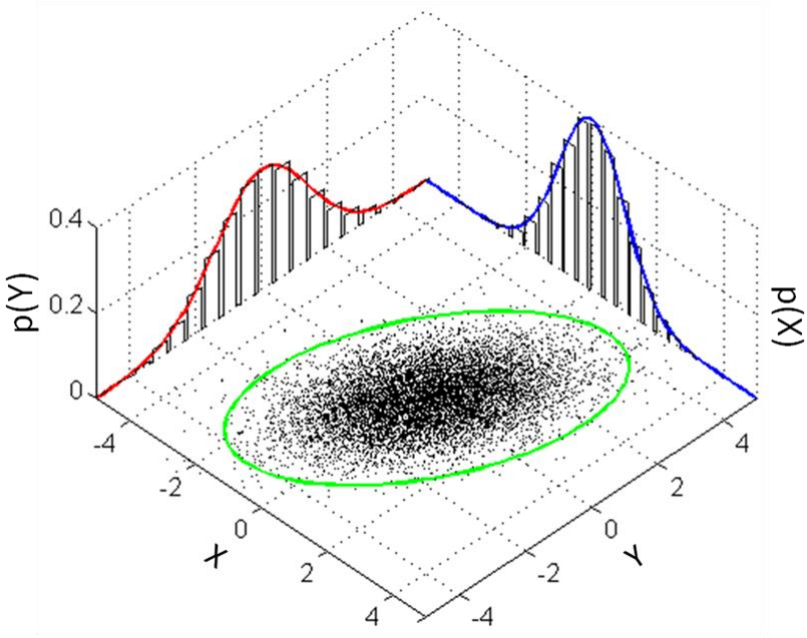


Figure 4: A sample from a joint probability distribution (bivariate normal). The ellipse shows a joint confidence interval based on a normal approximation of the sample (here: 3 times the s.d.). The marginal density functions are shown on the axes. This graph is taken from Wikipedia: https://commons.wikimedia.org/wiki/File:Multivariate_normal_sample.svg.

2 The statistical details

The frequentist view that we focus on revolves around the likelihood function and Wilk's theorem (as commonly applied in the likelihood-ratio test), and on constructing CIs by profiling the likelihood. To start with, our general error model is the likelihood function. The likelihood ($\mathcal{L}$) of a set of model parameters (θ), given a data set (Y), is the probability (P) to obtain the data set if that parameter set would have been the correct one (and the model is true). In math (the vertical lines should be read as 'given'):

$$\mathcal{L}(\theta|Y) = P(Y|\theta) \tag{2}$$

In the frequentist perspective, the data follow a probability distribution; each time we perform the same experiment, the outcome is slightly different, and we assume that this is caused by random variation. The parameters do not follow a probability distribution: they are fixed but unknown. Optimisation of the likelihood function can be written as:

$$\mathcal{L}(\hat{\theta}|Y) = \max_{\theta} \mathcal{L}(\theta|Y) \tag{3}$$

$$\hat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta|Y) \tag{4}$$

Where $\hat{\theta}$ is the parameter set that yields the highest value of the likelihood function (the best-fit parameter set).

2.1 A simple example with one parameter

To illustrate the likelihood function and its use with an example, let's toss a coin a number of times and count the number of 'heads' (that we will call a success). Suppose we find 20 successes out of 50 trials. What is the probability to find this particular outcome $Y = 20$? That depends on the probability p of 'heads' that we don't know (or at least, that we don't want to make assumptions about). Calculating the probability of the outcome is simple in this case, as we can use the binomial distribution:

$$P(Y = 20|p) = \binom{50}{20} p^{20}(1-p)^{50-20} \tag{5}$$

This probability is the likelihood $\mathcal{L}(p|Y = 20)$ that we are looking for; the likelihood is a function of the parameter p. We can easily calculate this likelihood for large range of values of the parameter p and look for the highest likelihood (Fig. 5). That value of the parameter then gets a hat as in $\hat{p}$. Not surprisingly, $\hat{p} = 0.4$ in this case. This is the parameter value that leads to the highest probability of finding this particular outcome $Y = 20$.

Next, we can calculate the CI of this parameter, given the data set. To this end, we can invoke a likelihood-ratio test. This is a general test to compare the fit of two nested models (i.e., models that only differ by fixing one or more parameters compared to the 'full' model) on the same data set, by taking the ratio of their likelihood values. How can

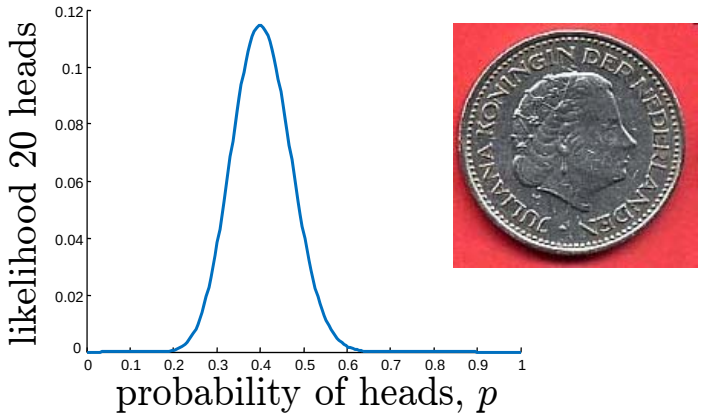


Figure 5: The likelihood function for the probability of the coin when observing 20 times heads in 50 throws.

we use this test to create a CI on the model parameter p? We can define the CI of p as the entire set of values for p that would not be rejected in a likelihood-ratio test. The two models that we compare are the 'full' model (with p as free parameter) to a 'reduced' model (where p is fixed). We can thus calculate the likelihood for a large number of values of p and compare it to the highest-possible likelihood (as generated by the best fitting parameter value $\hat{p} = 0.4$)

The ratio of two likelihoods (and thus the difference in two log-likelihoods) from nested models can be tested for significance; whether the fits of the models are different enough to yield a low probability of this difference happening purely by chance. To do this, we can use an 'asymptotic property' of the likelihood ratio, which means that it becomes a better approximation of reality, the larger the number of observations. Asymptotically, two times the difference in log-likelihood ℓ will follow a χ^2-distribution with as degrees of freedom (df) the number of free parameters in which the two models differ (in our case $df = 1$).[1] Using the natural logarithm of the likelihood ($\ell = \ln \mathcal{L}$):

$$2\ (\ell(\theta|Y) - \ell(\theta_1|Y)) \sim \chi^2_{df,1-\alpha} \tag{6}$$

We compare the full parameter set θ, and a reduced parameter set θ_1 (which in our simple example will be an empty set). When two times the difference between the two likelihoods exceeds the critical value of the χ^2-distribution, the zero hypothesis ('the fits are equally good') is rejected. Even though we are far away from an infinite number of observations with our data sets, the χ^2-distribution is a useful approximation for small data sets (see [16]). Note that the quantity $\ell(\theta|Y) - \ell(\theta_1|Y)$ is referred to here as the minus-log-likelihood ratio, MLLR (the log-ratio of two likelihoods is the difference in log-likelihoods).

Using Eq. 5 and 6, we can easily calculate the log-likelihood ratio for a range of values of p (Fig. 6). From this plot (or in this case even by direct calculation), we can find the values of p for which two times the likelihood ratio yields exactly the χ^2-criterion. There will be two of these values: the likelihood will get lower (and the ratio, as expressed in

[1] This is known as Wilks' theorem: https://en.wikipedia.org/wiki/Wilks%27_theorem.

Eq. 6, higher) on either side of $\hat{p}$. Will there be more than two of these crossings? Not in this case, but that is certainly possible in more complex models. The CI now consists of the set of all parameter values p for which the log-ratio is below the critical value, i.e., the parameter values between the two points where the log-ratio crosses the critical value.

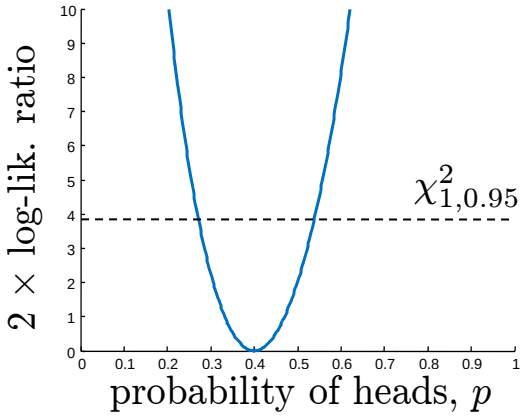


Figure 6: Plotting two times the log-likelihood ratio versus the probability of heads p. For the best-fitting parameter value, the ratio is one, and the log-ratio thus zero. Where the log-ratio exceeds the critical value of the χ^2-distribution, the values of p are rejected. The parameter set below the criterion is the CI (0.27-0.54, which includes the possibility that the coin is fair).

2.2 Extending to two parameters

How does this work out for a two-parameter model like the TK model we started out with? Firstly, the likelihood function will be a different one: rather than using the binomial distribution, we will likely invoke a normal distribution for the residuals (the difference between model and observation). Secondly, it is somewhat more complicated to construct CIs on the model parameters. For one free model parameter, we could simply calculate the log-likelihood ratio for a range of values of the model parameters (Fig. 6). With two parameters, we can can use a range for parameter 1, but what to do with parameter 2? The solution is to always use the best-fitting value for parameter 2, given the fixed value for parameter 1. In other words: we run through the range of parameter 1, and at each point for parameter 1, fit parameter 2 while keeping parameter 1 fixed. In mathematical terms, we can divide the total parameter vector θ into one or more parameters that we are interested in (θ_1), and 'nuisance parameters' that we are not (θ_2). The profile likelihood (ℓ_p) of a reduced parameter set (θ_1) is obtained by maximising over the nuisance parameters:

$$\ell_p(\theta_1) = \max_{\theta_2} \ell(\theta|Y) \tag{7}$$

We can plot the result as in Figure 6, and call this the profile likelihood of parameter 1. The curve is now the best value of the likelihood-ratio, given that we fix parameter 1 to the value on the x-axis. Each point on the line thus constitutes a likelihood-ratio test,

comparing the fit of the complete model (with two free parameters) to a reduced, nested, model (the model with parameter 1 fixed to the value on the x-axis). The difference in free parameters between both models is one, so for the critical value of the χ^2-distribution, we again need to use $df = 1$. After doing this for parameter 1, we can repeat the same process for parameter 2; profiling is done for each parameter separately to construct their CIs.

How does this profiling relate to our 2-D parameter-space plot of Figure 3? In Figure 3, we plotted 'lines of equal SSQ'. Now we moved to the likelihood as our goodness-of-fit measure, so we can plot lines of equal likelihood, or equal likelihood-ratio (relative to the best value), instead. Figure 7 shows equal-likelihood-ratio ellipses plotted at the likelihood ratio corresponding to critical value of the χ^2-distribution with $df = 1$ and $df = 2$.

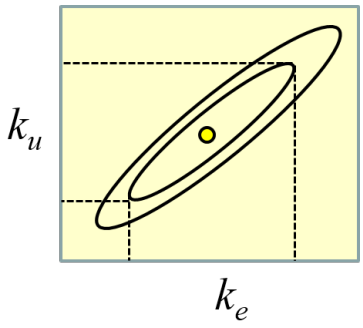


Figure 7: Lines of 'equal likelihood-ratio' plotted around the best-fit parameter combination (yellow point). The outer ellipse is the critical value of the χ^2-distribution with $df = 2$, the inner ellipse for $df = 1$. The edges of the inner ellipse indicate the CIs for the parameters.

Inside the inner ellipse at $df = 1$ are all the parameter sets that belong to the single-parameter CIs for parameter k_u and k_e: if we fix one of the parameters to a value outside of its CI, we will always reject the fit in a likelihood-ratio test at $df = 1$, no matter what the value of the other parameter. The outer ellipse at $df = 2$ is the joint CI for the two model parameters. If we fix two parameters to any point in this ellipse, we won't reject the fit. This ellipse is wider as we fix two parameters, rather than one, and we therefore need to accept a worse fit. The joint interval is not particularly useful, in my opinion, so we'll focus on the inner rim.

It is good to realise that Figure 7 is a 2-D representation of a 3-D landscape. By plotting lines of equal height, the plot is easier to interpret than the full 3-D version. The profile likelihoods are also 2-D versions of this 3-D landscape, but projected in a different manner: here we can see the height of the landscape (the likelihood ratio) but only one of the model's parameters. This is illustrated in Figure 8. All points in the ellipse for $df = 1$ (marked green) are somewhere in the green parts of the profile likelihoods. The profile likelihood itself (the curve) marks the *best* likelihood value (lowest likelihood ratio) to be found for each parameter value on the x-axis. It is expressed relative to the best value, which is plotted as zero; the quantity on the y-axis is two times the minus log-likelihood ratio (MLLR) as defined in Eq. 6. Apart from the parameter sets on the profile curve,

there also exist sets with worse values of the MLLR (where the other parameter is not at its optimal value), and they are somewhere above the profile curve. This is different from the one-parameter case in Figure 6: when there is only one model parameter, all likelihood ratios must lie *on* the curve (which therefore is a degenerative case).

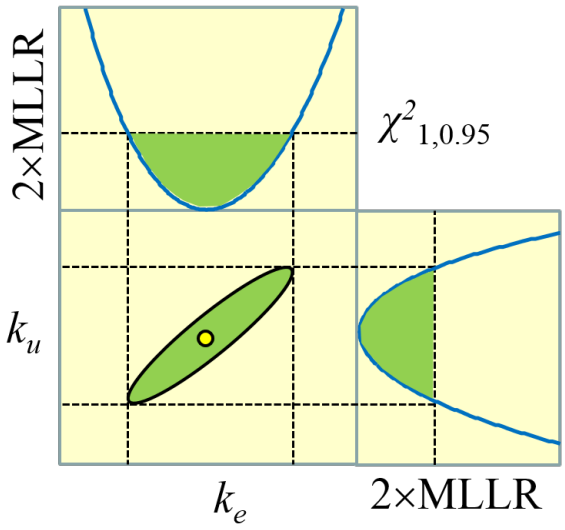


Figure 8: Relation between the parameter-space plot (center panel) and the profile likelihoods (top and right panel). All three panels are projections of a 3-D landscape in two dimensions. The y-axis of the profiles is two times the minus log-likelihood ratio (MLLR), which can be compared to a critical value of the χ^2-distribution (Eq. 6). The green area contains the same parameter sets in all panels.

2.3 Profile likelihood for model predictions

In the previous section, we saw how profiling of the likelihood function was useful to construct CIs on model parameters. However, we also would like to have CIs on model predictions: model outputs for situations where we have no observations (or where we have observations, but did not use them to calibrate the model). For example, we could calibrate our TK model to an experiment at constant exposure, and use the calibrated model to predict the internal concentration over time in a scenario with time-varying exposure. Obviously, the uncertainty in the calibrated model parameters will affect the certainty of our model predictions. How can we 'propagate' the errors from the calibration to the model predictions?

The answer is, again, profiling of the likelihood function. However, instead of fixing a model parameter to a range of values, we now fix a model *prediction* to a range of values. For example, with our calibrated TK model, we can predict the internal concentration at $t = 10$ days under a certain time-varying exposure scenario. The predicted internal concentration is on the x-axis of the profile plot (Fig. 9). We fix the prediction, and fit all model parameters again (to the calibration data set at constant exposure). This

requires constrained optimisation: fitting the model parameters under the constraint that a model prediction following from these parameters must have a certain value. This single constraint means that the we again have to use the critical value from the χ^2-distribution with $df = 1$.

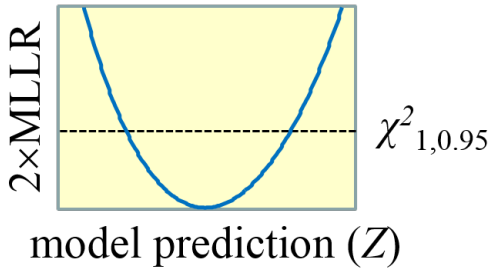


Figure 9: Profile likelihood for a model prediction. All model parameters are fitted on the calibration data set, under the constraint that the prediction must have the value at the x-axis. The y-axis of the profile is two times the minus log-likelihood ratio (MLLR), which can be compared to a critical value of the χ^2-distribution (Eq. 6).

In mathematical terms, we have to consider the prediction profile likelihood. Since we are interested in the model prediction (Z), the entire parameter set θ is the nuisance and needs to be profiled out. What we then obtain is (shorthand notation):

$$\ell_p(Z) = \max_{\theta|f(\theta)=Z} \ell(\theta|Y) \tag{8}$$

We thus find the parameter set θ to maximise the likelihood (as in Eq. 4) with a constraint: some function of the model parameters $f(\theta)$ should have the prediction Z as its output.

2.4 Practical issues and the sampling approach

In the previous sections, I outlined a framework for model optimisation and construction of CIs on model parameters and model predictions. All that is needed is an appropriate likelihood function and optimisation routines. We need optimisation to find the best-fitting parameter set, and after that, to construct CIs (profiling is basically repeatedly running an optimisation routine with one parameter fixed or with a single constraint). However, we generally need to do *a lot* of optimisations. If we want to make a profile with 50 points on the x-axis, we need to do 50 optimisations, for each model parameter and for each model prediction. Each optimisation requires starting values and runs the risk of ending up in a local minimum if the starting values are not well chosen. This risk can be reduced by using more-robust global methods, but the demands on calculation time will rapidly become prohibitive. Furthermore, it is inefficient to run so many optimisations independently: evaluating the likelihood at so many points in parameter space, and throwing the result away before moving to a next optimisation.

We could, in principle, simply evaluate the likelihood function at a lot of points in parameter space, e.g., a regularly-spaced grid. In a 2-D parameter space, and when we

can limit the parameter ranges to search, this would be feasible. When we have enough points, we can get a good approximation of the green area in Figure 8: the set of all parameter combinations that are not rejected in a likelihood-ratio test with one degree of freedom. The green area in the parameter-space plot is the same parameter set as the green section plotted in the likelihood profiles for the single parameters. Thus, from a good approximation of the green area of parameter space, we automatically get a good approximation of the CIs on the model parameters.

The profile likelihoods in Figure 8 are in fact the lower edge of a 3-D parameter landscape, viewed from one side (a 2-D projection). Can we also interpret the prediction profile of Figure 9 as a projection from this landscape? We would need to extend the landscape with a fourth dimension: each parameter combination has an associated likelihood, but also an associated model prediction. This is thus a 4-D landscape; impossible to plot in a simple way, but the principles remain the same. We can project this landscape onto a 2-D plot, with the prediction on the x-axis and the MLLR on the y-axis. The prediction profile of Figure 9 was built from (constrained) optimisation, but in fact it is the lower edge of this 2-D projection. The green area in Figure 8 thus also corresponds to the green area in the prediction plot of Figure 10.

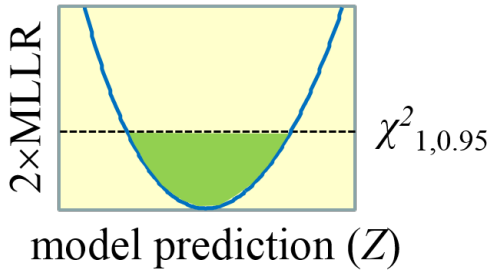


Figure 10: Profile likelihood for a model prediction. The parameter sets in the green area in Figure 8 also appear in the green area here. The y-axis of the profile is two times the minus log-likelihood ratio (MLLR), which can be compared to a critical value of the χ^2-distribution (Eq. 6).

This gives us an efficient strategy for model optimisation and construction of CIs on parameters and model predictions. If we have a sufficiently detailed sample from the 'relevant part' of parameter space, we can calculate a likelihood at each point, and predictions at each point, and make projections to approximate the profiles that we are interested in. For the predictions, we will not generally be interested in the entire prediction profile; we just need CIs. Therefore, we can take a short-cut. The CI on the model prediction is given by the points at which the prediction profile crosses the χ^2-criterion. Therefore, the only relevant part of parameter space is the set of parameter combinations that sit at the χ^2-criterion. Since we will have a discrete sample of finite size, we will need to consider the parameter sets close to this criterion (no set will be exactly on it). So, instead of calculating the model prediction for each possible parameter set, we can focus on the parameter sets that yield a likelihood ratio close to the χ^2-criterion, make predictions from them, and take

the minimum and maximum of these predictions as the CI. This is illustrated in Figure 11: we only need to make predictions for the parameter sets in the grey band around the χ^2-criterion to create CIs on the model prediction.

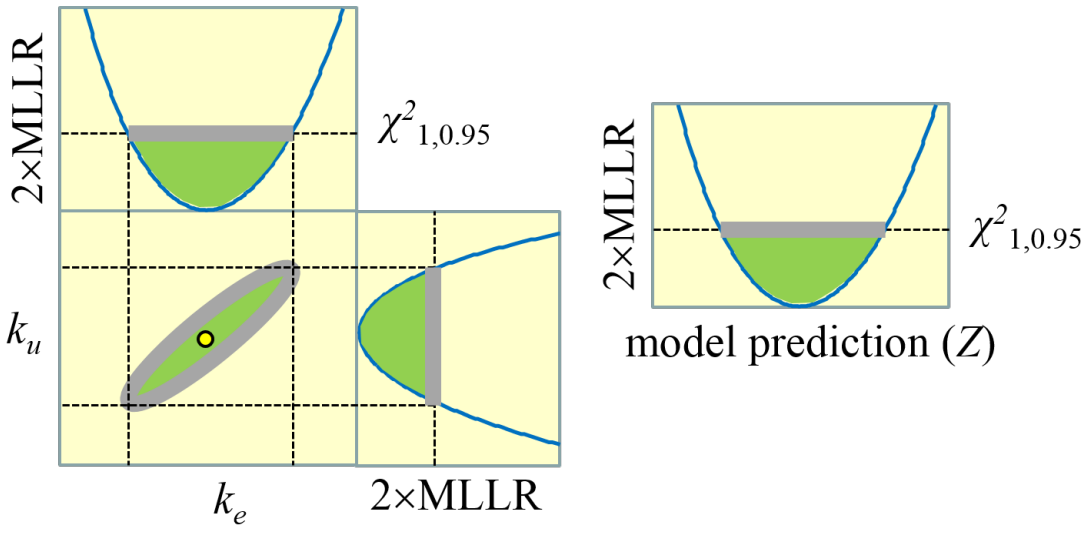


Figure 11: Parameter space, with the profile likelihoods for the parameters and a model prediction. The green area contains the same parameter sets in all panels, and the same holds for the grey band.

The tricky part is to get a good sample from parameter space, with the best-fitting parameter set and sufficient detail in the surroundings of this point. This task becomes rapidly more complex with an increase in number of free parameters (which implies more dimensions of parameter space). If we can decrease the range of each parameter, this task becomes more manageable. For example, the elimination rate in the TK model (k_e) will always be between zero and infinity, which is an impossible space to sample. However, very high values are not so interesting: they imply immediate steady state. The difference in model behaviour between a high value of k_e and 10 times that value will be irrelevant. Similarly, very low values are not so interesting: they imply that there is no steady state on a relevant time scale (the internal concentration just increases linearly over time). Therefore, based on our time scale of interest, we should be able to set a range of 3-4 orders of magnitude for where the interesting values lie. Such a range can be efficiently sampled (on log-scale).

With some careful reasoning, we can set such ranges for each model parameter and thereby limit the parameter space to search. However, for models with more than two parameters we enter the 'curse of dimensionality', as parameter space becomes a horribly large (hyper-)volume. Therefore, we need to come up with a more efficient approach than simply taking random points or a regular grid across parameter space.

2.5 Additional considerations and further reading

If we have an infinitely large data set ('asymptotically'), our iso-likelihood lines will be perfectly elliptical, and our profiles all perfect parabolas. However, in practical cases, we will often encounter more strangely shaped parameter landscapes. There may even be disconnected 'islands of good fit'. This is one of the reasons why it is a good idea to map this landscape, and to base CIs on the shape of the landscape, rather than assuming that is perfectly asymptotic (which is often done in statistics). However, the use of the χ^2-distribution for assessing the likelihood ratio (Eq. 6) also rest on asymptotic theory. Fortunately, this call on asymptotic theory is not as demanding as the previous one. So, constructing profile likelihoods and evaluating them against a χ^2-criterion is generally a good idea [16]. However, for small data sets, the coverage of the CI may be somewhat more or less than 95%.

Application of likelihood-based inference methods obviously requires an appropriate likelihood function. For GUTS [9], the likelihood function can be derived from the multinomial distribution (or the equivalent conditional binomial), which is a good match to the problem. However, for DEBtox [11, 8], we are usually in a situation that growth and/or reproduction is followed on the same group of animals over time. In that case, we have dependent observations. Note that, in GUTS, we also follow a group of animals over time, but for each animal there is only one fundamental observation: the interval in which it dies. This observation can be assumed to be independent (the death of one individual should not affect the death probability of another). An appropriate approach for dependent observations is not so straightforward, especially because the 'errors' that we observe are not caused by random measurement errors but largely by inter-individual variation (see also [6]). For DEBtox, I am not aware of suitable likelihood functions for dependent observations; in practice, everybody assumes independent distributions for the errors (usually normal, in some cases after transformation). In my opinion, it is still a good idea to use this likelihood function, and to make CIs, although the interpretation needs to be more qualitative.

More technical information about calculation of the likelihood and construction of confidence intervals can be found in textbooks [16] or the open literature. I especially recommend the following papers for more background on likelihood profiling: [14, 12, 17]. Regarding the 'prediction profile likelihood', see [13, 12] (these papers have helped me tremendously to make the link between parameter landscapes and CIs on model predictions).

3 Short description of the reduced GUTS models

The statistical framework (Section 2) and the specific algorithm (Section 4) were designed in a project developing a support tool for reduced GUTS models (see http://openguts.info/about.html). GUTS stands for the General Unified Threshold model for Survival, first published in 2011 [9], and described in detail in a freely-downloadable e-book [10]. However, it is good to stress that the statistical framework can, in principle, be used with any model (as long as a likelihood function is provided). The algorithm contains very few GUTS-specific elements, only the choices the search ranges of each parameter and the check on 'slow kinetics'.

To make this paper (and especially the case study) readable for those without knowledge of GUTS, a short model description is provided here. GUTS is a simple TKTD framework for effects on the endpoint survival (and other all-or-nothing events that can be treated as irreversible chance events). The openGUTS software tool implements the two simplest models from this framework (reduced pure SD and reduced pure IT), so this description will be limited to these models (see Fig. 12). The assumptions underlying these models are:

1. A chemical first needs to be taken up into the body (toxicokinetics), and then causes damage that can be repaired (damage dynamics). In the reduced GUTS models, these two processes are lumped: damage is directly linked to the external concentration and follows one-compartment dynamics with first-order kinetics. Damage accrual is proportional to the external concentration, and damage repair is proportional to the level of damage. Damage is kept abstract, and we follow the scaled damage. This implies that damage dynamics is specified by only a single parameter: the dominant rate constant k_d. The damage level is driving the toxic effect.

2. Death is a chance process. For the stochastic death (SD) model, all individuals are identical, and the instantaneous probability to die (the hazard rate) is proportional to the damage level above a threshold value m_w (the proportionality is the killing rate b_w). For the individual tolerance (IT) model, death is immediate when damage exceeds a threshold, but this threshold follows a log-logistic frequency distribution in the population (with median m_w and spread factor F_s).

3. Background mortality is independent of the mortality caused by the toxicant, and is taken as a constant hazard rate.

4. The likelihood function follows from the assumption that death is a chance process that occurs only once in the lifetime of an individual, and the chance to die is for each individual independent.

Translated into model equations, damage dynamics is given by the following differential equation:

$$\frac{dD_w}{dt} = k_d(C_w - D_w) \quad \text{with } D_w(0) = 0 \tag{9}$$

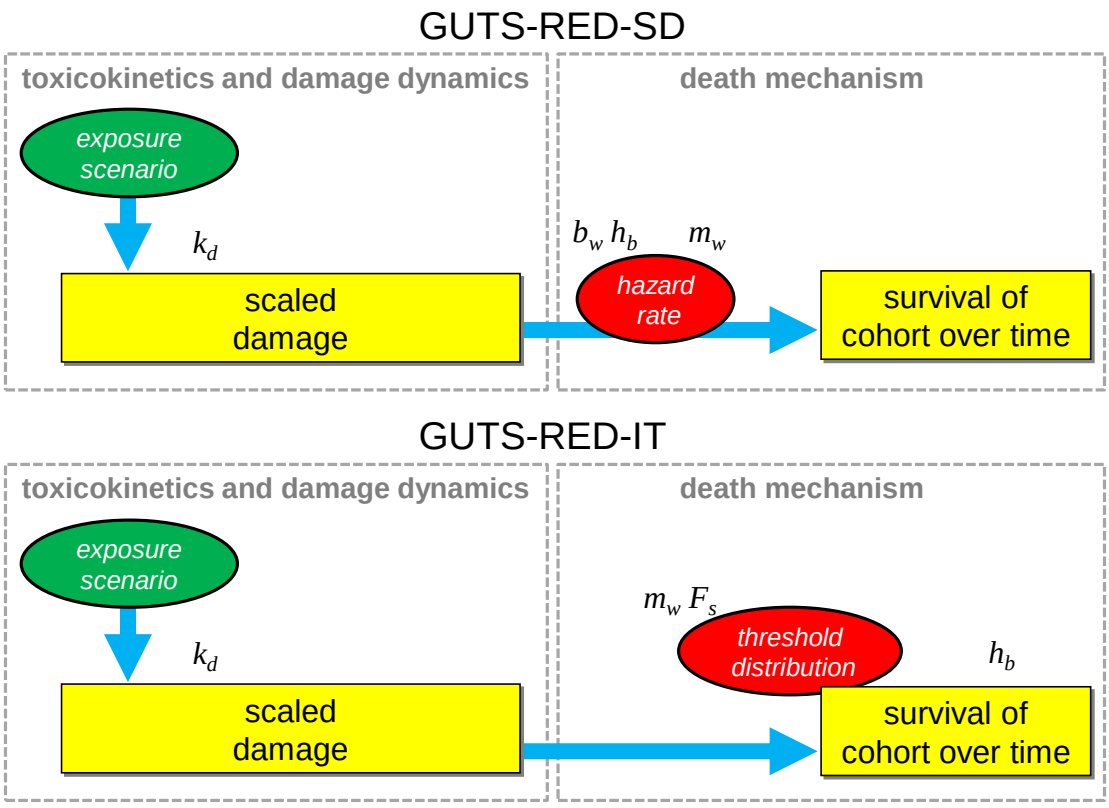


Figure 12: Schematic representation of the reduced GUTS models for pure SD and pure IT, including the parameter symbols.

This equation is used for the SD model as well as the IT model. Here, D_w is the scaled damage (a state variable), C_w is the external concentration (as function of time; a forcing), and k_d is the dominant rate constant (a model parameter). Note that damage has the dimensions of an external concentration, hence the designation scaled damage and the subscript w (it is referenced to the external concentration, here taken to be in water). The openGUTS software tool applies analytical solutions for this equation, which is feasible as the exposure profile must be entered as a series of linear (or instant) changes over time. The analytical solution for a linear change in forcing is applied sequentially over the exposure profile.

For SD, the hazard rate due to chemical stress (h_c) is calculated from damage D_w following a linear-with-threshold relationship:

$$h_c = b_w \max(0, D_w - m_w) \tag{10}$$

Where m_w is the threshold for effects (a model parameter), and b_w the proportionality constant or killing rate. Note the subscript w as these parameters are also referenced to a water phase. The hazard rate is translated into a survival probability S_c by integration over time:

$$S_c = \exp\left(-\int_0^t h_c(\tau)d\tau\right) \tag{11}$$

Working with hazard rates in this fashion is a standard statistical approach to deal with chance events over time. The openGUTS tool applies an analytical solution for this integration when the exposure concentration is constant over time, but trapezium-rule integration otherwise.

For the IT model, we first need to establish the maximum value of the scaled damage (D_{wm}) that has occurred over time. This is needed because dead animals will remain dead, even when damage decreases again later in time. No integration is needed for this model, as death is immediate when damage exceeds a threshold. The threshold is assumed to follow a log-logistic distribution in the population:

$$S_c = \frac{1}{1 + (D_{wm}/m_w)^\beta} \quad \text{with } \beta = \frac{\ln 39}{\ln F_s} \tag{12}$$

The threshold distribution is specified by a median m_w (model parameter) and a factor of spread F_s (model parameter). The spread factor is a practical measure for log-symmetrical distributions: 95% of the threshold values are within a factor of F_s from the median. This parameter has a simple link to the more familiar β of the log-logistic distribution.

For both model cases, when S_c is established, it needs to be multiplied with the background survival probability. As we are usually dealing with short-term experimental data, background mortality is not due to ageing, and can usually be well represented by a constant hazard rate (representing handling effects and other random causes of death):

$$S = S_c \times \exp(-h_b t) \tag{13}$$

Where h_b is the background hazard rate (model parameter), and S is the total survival probability (model output).

The likelihood function is based on the multinomial distribution. This is the appropriate distribution for the data, as the data are for a single discrete (irreversible) event in the individual's life. The log-likelihood function for a parameter set θ and a data set X is (first considering a single treatment C_w):

$$\ell(\theta|X) = \sum_i x_i \ln p_i(C_w, \theta) \tag{14}$$

The individual observations for each time interval between observations i are denoted as x_i (the numbers of deaths in interval i). The (unconditional) death probability for each interval is p_i. The unconditional death probabilities can be easily calculated from the GUTS predictions on survival probability at the start of each interval S_i, and the observed deaths follow from the observed number of survivors at the start of each interval y_i:

$$p_i = S_i(C_w, \theta) - S_{i+1}(C_w, \theta) \tag{15}$$

$$x_i = y_i - y_{i+1} \tag{16}$$

Note that the last interval to consider is from the end of the experimental test to infinity, as the probabilities p_i over time need to sum to one (to satisfy the conditions for a multinomial distribution). For the last interval $S_{i+1} = 0$ and $y_{i+1} = 0$. Since we will usually have multiple treatments in a data set, the log-likelihood $\ell(\theta|X)$ needs to be summed over all treatments (the treatments are taken to be independent).

4 The parameter-space explorer algorithm

4.1 Introduction

The parameter-space explorer algorithm was developed by DEBtox Research as part of the openGUTS project to find and map the relevant part of parameter space for reduced GUTS models, given a specific data set. The relevant part of parameter space is the (hyper-)volume that contains the best-fit parameter set and all sets that are within the cut-off criterion based on the critical value of the χ^2-distribution with one degree of freedom. To add some robustness, and to provide more insight into the shape of the parameter landscape, a larger volume will be mapped: by default all sets within the χ^2-criterion with as degrees of freedom the total number of fitted parameters (this is the joint confidence region for all free parameters). The sample from parameter space is used to construct the CIs on model parameters and model predictions.

An important constraint in the openGUTS project was that the algorithm should operate fully automatic, without user interaction. This is only efficient when the volume of parameter space can be limited as much as possible. To this end, a series of heuristics was developed for the GUTS model, deriving ranges based on the data set, the foreseen extrapolations, and some general considerations (see the openGUTS design document on https://openguts.info/download.html). Another important requirement was robustness, as the openGUTS software is intended for use in a regulatory context. Since the algorithm covers a large part of parameter space, it is much more effective in locating the global optimum than local methods such as the Nelder-Mead simplex. However, to increase robustness, and to ensure proper sample coverage of parameter space, sampling is amended with optimisation and explicit likelihood profiling (sequential optimisation). It turns out that parameter space for GUTS problems can be very strangely shaped, which makes mapping difficult in such cases (see typical examples in the interpretation background document from http://openguts.info/download.html).

4.2 The algorithm

The framework to tackle the requirements listed above is based on sampling from parameter space. The algorithm in openGUTS combines grid search, a genetic algorithm, and likelihood profiling, to provide extreme robustness. The outline of the entire algorithm is shown in Figure 13. The general idea is to try a large population of candidate parameter sets, and select the most promising sets for a next round, where each set will be propagated with random mutations to a new set of candidates. The evaluated sets from the previous rounds do not need to be discarded: the only thing that counts is whether their likelihood is within a certain distance from the best-fitting set. However, as the best-fitting set is also subject to change with consecutive rounds, the confidence region (CR) will change as well (and a set that was acceptable in round i may not be anymore in round $i+1$).

In principle, we can keep all of the sets that are evaluated: their likelihood value does not change as the optimisation progresses. However, in view of memory use, the set is pruned every round: sets are removed that are too far outside of the 95% joint CR as

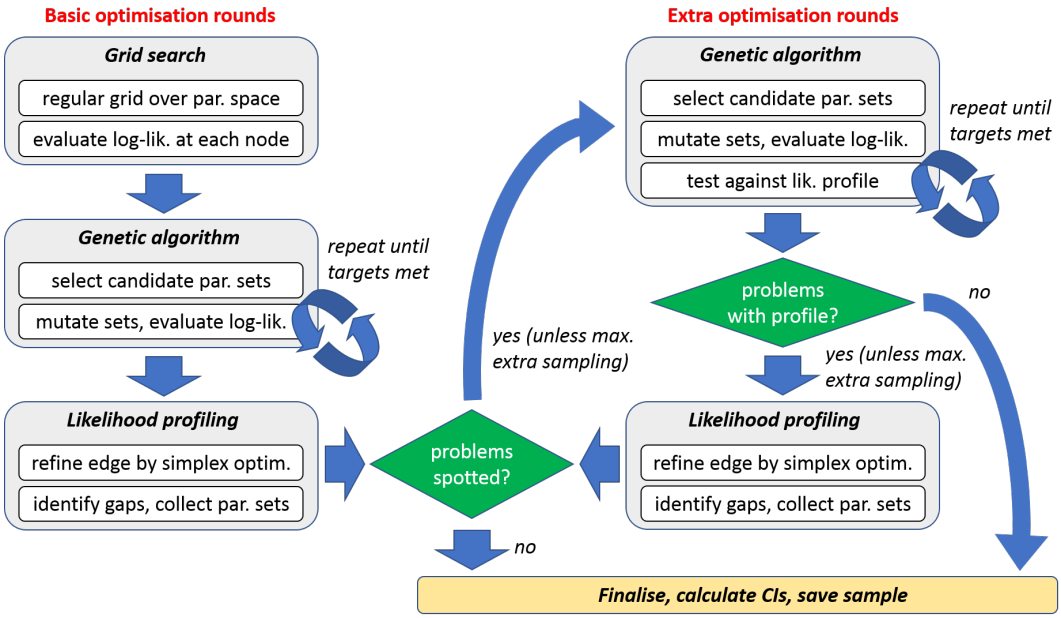


Figure 13: General flowchart for how the optimisaton algorithm operates. Well-behaved data sets will only require basic optimisation (left part of the scheme); the extra optimisation (right part of the scheme) is a fail-safe to take care of more problematic data sets. The genetic-algorithm step will be repeated several times until its 'targets' are met (minimum number of sets in inner and outer rim, and for the extra rounds also a check against the profile likelihood). At several points in the process, there is a check against a maximum number of iterations to avoid getting stuck.

established in that round. For propagating the uncertainty due to model predictions, we only need the sets within a smaller joint CR (based on $\chi^2_{df=1,\alpha=0.05}$). Nevertheless, for robustness (to make sure that local minima are not missed, and to still have a sample when the best fit improves) it is essential to let the algorithm generate a sample of the 95% joint CR, from which we can take a sub-set with smaller coverage later on.

The random mutation of parameter sets continues for several rounds until the targets are met (sufficient points in the sample, or maximum number of rounds reached). In principle, the sample can be used directly to approximate the CIs on the model parameters (as explained in the Section 2). However, to increase robustness, the algorithm uses explicit profiling to refine the edge of the sample. This implies repeated optimisations, using the sample to provide the starting values (which is robust and efficient). In most cases, this will only marginally improve the CIs on the model parameters. However, in cases where the parameter landscape is strangely shaped, sampling may not be efficient, and profiling will help reveal that.

These steps are the core of the algorithm, and are on the left side of Figure 13. However, after these steps, it is possible that some problems have been identified. For example, the profiling step may have located a better optimum, or there are gaps between the profile and the sample, or insufficient points in the sample. In those cases, additional rounds of

mutation are performed, focussing on the problematic parts of parameter space. And, if needed, a new round of profiling is initiated. These extra steps improve robustness, at the cost of a substantial calculation time.

In more detail, the algorithm operates following these steps:

Module 1. Grid search

1. Create a regular grid in parameter space, covering the ranges of the (log-transformed) parameter values. Ranges were established based on the time and concentration vectors of the data set, the foreseen extrapolations, and a set of heuristics. The grid thus has the same dimensions as the number of fitted parameters.

2. Evaluate all parameter sets from the grid: each parameter set now gets a minus-log-likelihood (MLL). This is not an optimisation, only a single evaluation. All parameter sets with their corresponding MLL are stored in the results matrix.

3. Find the best fit (lowest MLL) so far, and select the parameter sets whose MLLs are within a certain distance from the best fit. If this results in less than a minimum number of sets, use the minimum number of best ones instead. These parameter sets are collected in a matrix of new candidate sets.

Module 2. Genetic algorithm In this module, the same series of steps is being repeated for a number of subsequent rounds.

1. For each parameter set in the candidate matrix, create a number of new parameter sets using a random mutation of the parameter values. For each parameter, a random number is added or subtracted between zero and a fraction of the initial parameter-grid spacing. This fraction will decrease with each round, such that the sample can contract.

2. The mutated parameter sets from the previous step are evaluated against the data set (again, no optimisation), yielding a MLL, which is stored with the parameter set in the results matrix.

3. Find the best fit (lowest MLL) so far, and perform a quick and dirty simplex optimisation to improve upon the optimum achieved so far. Add this optimised parameter set to the results matrix.

4. Check for signs of whether a restart of the algorithm is needed (when 'slow kinetics' is indicated, as explained in the case study in the main text and Section 7). This check is only performed under certain conditions (when the threshold is fitted on normal scale and the dominant rate constant on log-scale).

5. See how many parameter sets are within the joint CR (MLLR within a specific criterion, depending on the number of free parameters) and the single-parameter region (MLLR within the criterion using $\chi^2_{df=1,\alpha=0.05}$). If it is enough, or when we have reached a maximum number of rounds), jump to Module 3.

6. Prepare for a new round. Based on the results so far, select an optimal set of candidate parameters to continue with. In general, we will continue with promising new sets generated in the last round only, to avoid resampling from the same points over and over again. However, if the number of sets is too small, sets from all previous rounds will be included as well. In special situations, a more specific candidate matrix is defined. When we have sufficient points in the total cloud, but not enough in the inner rim (the single-parameter region), only sets around the inner rim are taken. Check how many new mutations will be tried in the next round, and if it is very high, reduce it. If few good values are found, increase it. Return back to step 1 of this module.

Module 3. Likelihood profiling

1. Run a regular simplex optimisation, starting from the best set established so far in the results matrix, to make sure we are in the optimum.

2. Refine the edges of the cloud for a single-parameter with profiling (sequential optimisations along a parameter's range), using the sample as basis. This yields a very robust profile likelihood. If a better optimum is located, run a regular simplex optimisation to improve it.

3. The profiling collects the points where the profile (the optimised log-likelihood) is substantially better than the best sample point in that range of the parameter; this indicates an area where sampling has been poor. Furthermore, it collects the points where the profile extends beyond the range of sample points. If the profiling has located a much better optimum, this shifts the sample, and there may not be enough points in the inner rim anymore. If one of these situations happen, an additional series of sampling rounds is initiated, using the collected points as candidate sets. If no problems are identified, jump to Module 6.

Module 4. Extra rounds of sampling

1. Mutate candidate parameter sets randomly and calculate the associated MLL (again, no optimisation). Test the new total sample in relation to the profile likelihood. Collect points where the profile is much better than the sample, and flag when there are sample points that are below the profile line. These are the candidates for further sampling.

2. Decide whether to continue sampling. If there are insufficient sets in the inner rim, go back to step 1 of this module. If the previous step finds points where the profile is much better than the sample, use them in another round of sampling, also go back to step 1. If no problems are flagged, or the maximum extra sampling/profiling rounds has been reached, jump to Module 6.

Module 5. Extra round of profiling

1. When there are sample points that are (non-trivially) below the profile line, do another profiling step. If a better optimum is located, run a regular simplex optimisation to improve it. If this results in flagged points where the profile is much lower than the sample, use these points as candidate sets in another sampling round, and go back to Module 4.

Module 6. Finishing up and reporting

1. Calculate CIs from the profiles. Check whether any CIs are open on one side and mark them. Check whether CIs run into bounds. Also check whether the CI is a broken set. The sets from the profile are added to the total results matrix as well. This helps in rather extreme cases where the mutation algorithm had difficulties to sample in a relevant region of parameter space.

2. Display results on screen. Note that for broken CIs, only the outer edges are reported in openGUTS: the min and max of all points where the likelihood-ratio crosses the χ^2-criterion.

3. Save the calibration results to file to use for all model predictions that require CIs on model curves.

4.3 Some details of Module 2

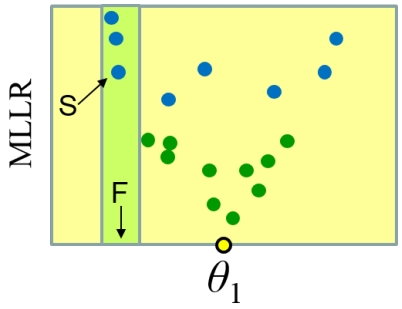
In Module 2, the settings for the algorithm will change with each round of the analysis, and will be modified based on how many good points are found. The cut-off criterion for continuation of candidate sets will decrease with each round to end up at the final value (based on the χ^2-criterion, 95% confidence, with as *df* the number of free parameters). The number of new tries per parameter set decreases as well: initially, we need many tries to get a robust coverage of parameter space, but when we already have a large number of sets within the joint CR, we can use less tries in subsequent rounds (otherwise, we end up with much more values than needed). The maximum mutation jump distance will also decrease with subsequent rounds, starting at 1 (so the maximum mutation is the same as the grid spacing used). All these settings have been tuned to obtain a good balance between robustness and speed of contraction of the sampling cloud for a wide range of relevant example data sets (with robustness more important than speed).

Step 6 of Module 2 is the crucial one, and involves a number of criteria to select the candidate parameter sets and settings for the next round. In principle, only the new values tried in a round, that are within a certain criterion for this round, are propagated to the next round, to avoid mutating the same sets of parameters over and over in consecutive rounds. However, this set may be very large or very small, which requires some modifications of the set and/or of the number of new tries in the next round. The aim is to end up with an accepted parameter sample that is not too much more than the minimum size specified, and that provides good coverage of the relevant part of parameter space, for all possible data sets.

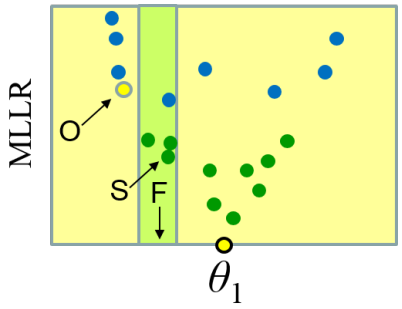
In Step 6 of Module 2, the algorithm will also check how many sets there are within a narrower CR (based on $\chi^2_{df=1,\alpha=0.05}$), the inner rim. The edges of this cloud mark the single-parameter CIs. We need to have sufficient sets within this inner cloud, as it is used to calculate the CIs on the single parameters, and because it is used to calculate intervals on model predictions. If there are sufficient accepted sets in the joint region, but insufficient sets in the inner region, the next round will continue with mutating candidate sets from the inner region.

4.4 Some details of Module 3

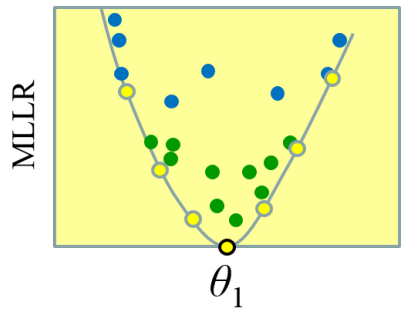
In profiling, each fitted model parameter is treated separately. The sample is viewed from the perspective of one parameter (θ_1), plotted on the x-axis, with the MLLR on the y-axis (so the best value is plotted at zero, and poorer fits at positive y-values). The parameter's relevant range (as found so far) is divided into 50 slices. We start with the left-most slice (green). Parameter θ_1 is fixed to the middle of the slice (F), and the other parameter are fitted, using the best set from the sample in this slice (S) as starting values for the free parameters (excluding θ_1).



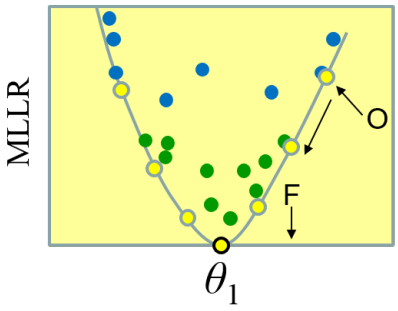
This yields an optimised point on the profile (O). Now we move to the next slice, and repeat the procedure: fix θ_1 to the middle of the slice (F), and select the best set from the sample in this slice (S). However, now we perform two rough simplex optimisations in each slice: one starting from the best-fitting set in this slice, and one starting from the optimised set in the previous slice (O). The best of these two optimisations is subsequently used in a regular simplex optimisation, yielding an optimised point for the new slice.



This procedure continues until we have covered all slices of the parameter's range, and we can draw a profile curve.



In principle, this should have been sufficient to yield a robust likelihood curve. However, in some very recalcitrant cases, profiling misses a branch in parameter space (it fails to spot a better optimum over a ridge in the landscape). As an extra safety measure, we repeat the procedure, going back from right to left, fixing θ_1 to the value of an optimised point (F) (which was also the middle of a slice), but optimising using the parameter values (for the free parameters) from the previous point in the profile (O).



When the edges of the resulting profile are not well above the cut-off criterion (based on $\chi^2_{df=1,\alpha=0.05}$), that means that the sample did not capture the inner rim well enough. In that case, the profiles are extended outwards until they are well above the criterion or until they hit a boundary. If we find a better optimum, refine it with another optimisation.

In the profiling process, candidate parameter sets are collected for additional mutation. These candidates are parameter sets in the sample where an optimised point (O) is much better than the best element of the sample in that slice (S). Both points will be selected as candidates for mutation. Additionally, optimised points will be collected where the profile is extended beyond the sample (this mainly happens when parameters run away to an upper or lower boundary of their range).

4.5 Some comments on Module 4/5

Module 4 and 5 are a safety measure. For friendly data sets, they will not generally be triggered, and if they are, the refinement will not be substantial. However, there are data

sets where the core sampling routine fails to sample a particular part of parameter space, or even (in very extreme cases) misses the global optimum. Profiling will indicate these issues, and the iterations between profiling and targeted mutation are effective to still obtain good coverage of the sample. Downside is increased calculation time and often a much larger sample in total. However, for application in risk assessment, robustness is more important than speed.

5 Demonstration for a simple case

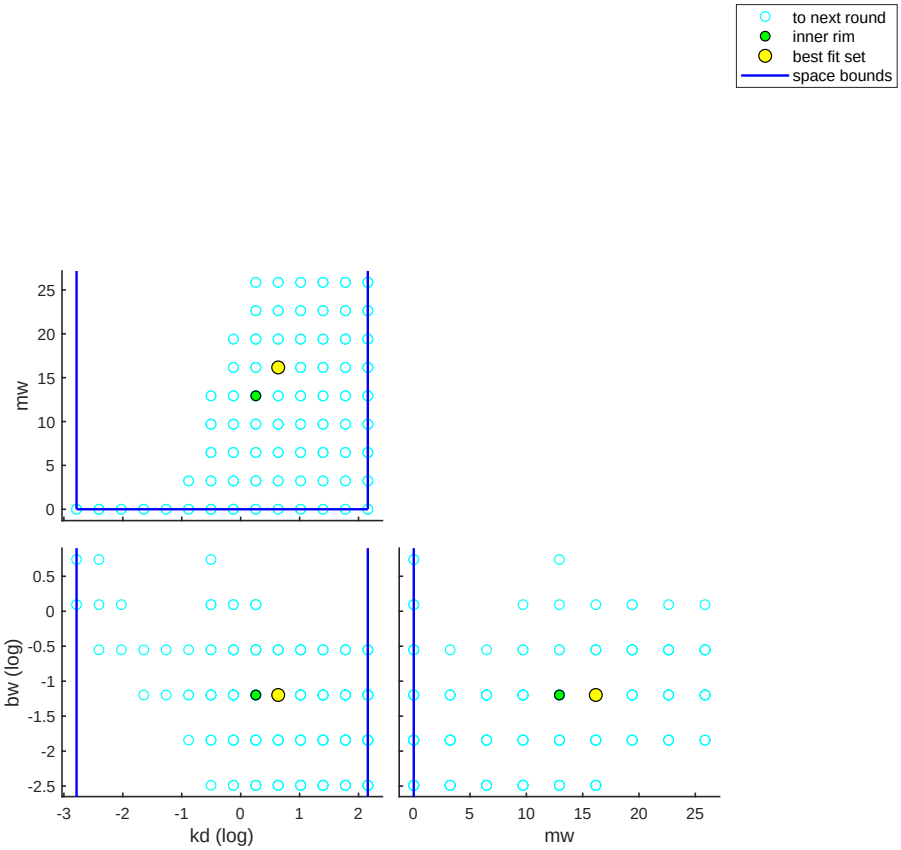
This chapter runs through a well-behaved case to illustrate how the main part of the algorithm operates in more than two dimensions. The plots are from the openGUTS Matlab version (http://openguts.info/download.html), using the case study for *Gammarus pulex* exposed to propiconazole (constant exposure) [15]. The GUTS model is fitted with three free parameters; background hazard is fixed to the control mortality. However, fitting the background hazard together with the other parameters is also possible. Parameter space is thus 3-D, and with the associated likelihood based on the data set, we get a 4-D landscape. The 3-D parameter space is plotted as three 2-D projections (as in Fig. 7), and the likelihood dimension is shown as different colours of the points (rather than the ellipses for the asymptotic case).

Round 1, Module 1. Grid search The algorithm starts from a regular grid, defined by the search ranges. Note that there are five parameters in the reduced GUTS models, but only three are fitted.

```
Settings for parameter search ranges:
=====================================================================
kd    bounds:   0.001641 -      143.8 1/d      fit: 1 (log)
mw    bounds:   0.002202 -      35.56 uM       fit: 1 (norm)
hb    bounds:    0.01307 -    0.01307 1/d      fit: 0 (norm)
bw    bounds:  0.0007332 -       9310 1/(uM d) fit: 1 (log)
Fs    bounds:          1 -          1 [-]      fit: 0 (norm)
=====================================================================

Starting round 1 with initial grid of 2016 parameter sets
```

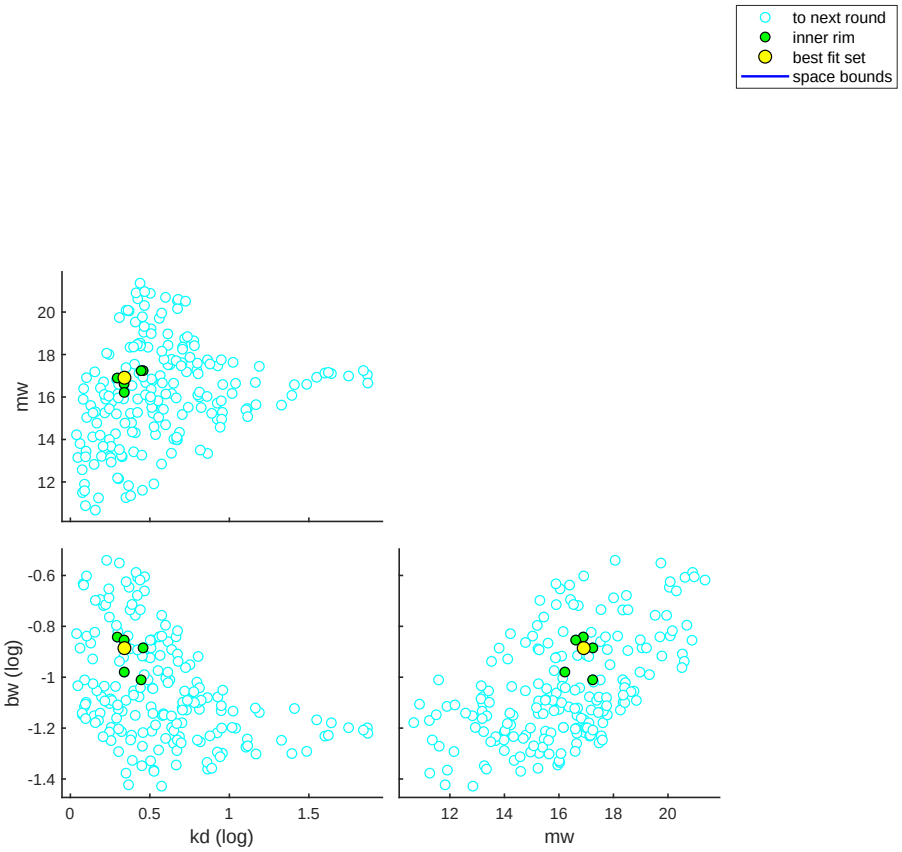
The algorithm calculates the likelihood at each point of a regular grid. The plot shows the best value so far (yellow), the points within the inner rim (based on $\chi^2_{df=1,\alpha=0.05}$, green), and the points that are 'good enough' to act as candidates for the next round (blue, unfilled). Note that the plot axes are scaled to focus on the relevant area of parameter space (covering the yellow point, and the green and blue points).



After this round, the algorithm gives the current status on screen, and reports what it will do in the next round:

```
  Status: best fit so far is (minloglik) 131.8625
Starting round 2, refining a selection of 200 parameter sets, with 60 tries each
```

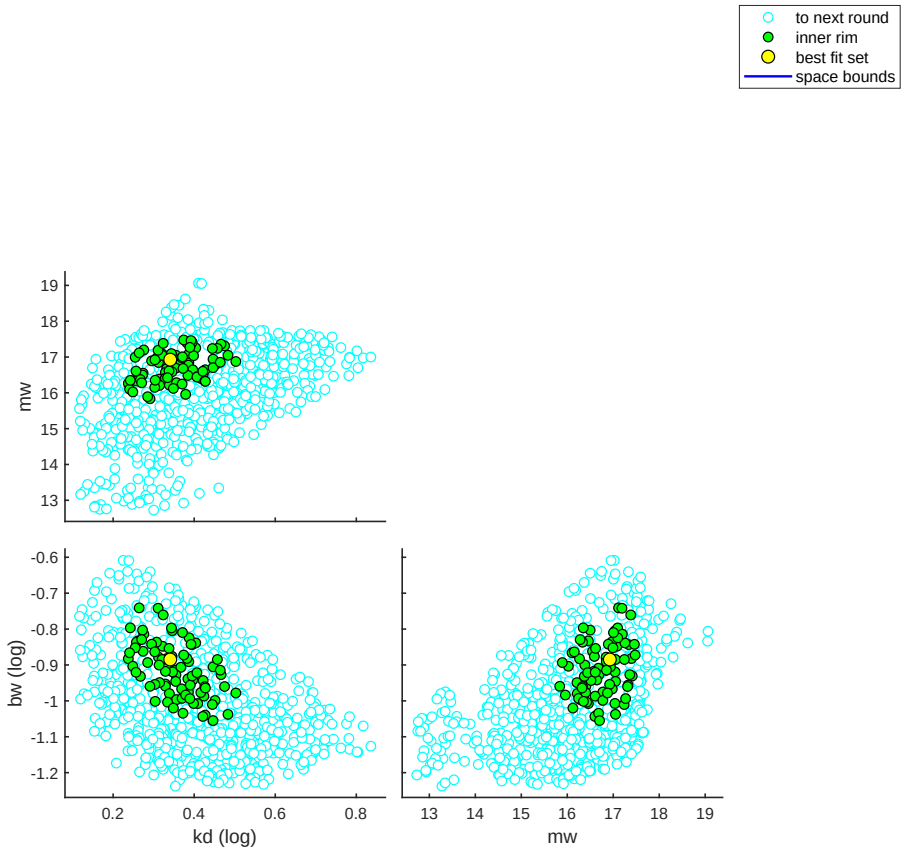
Round 2, Module 2. Genetic algorithm In the next round, the algorithm moves to Module 2, and starts to mutate the candidate parameter sets identified in the previous round. Afterwards, it plots a new picture of parameter space. After one round of mutations, parameter space is already starting to take shape.



A status update is printed on-screen. This shows that we only have very few points that are within the joint CI (based on $\chi^2_{df=3,\alpha=0.05}$) and even less within the inner rim used to construct CIs on model parameters and predictions (based on $\chi^2_{df=1,\alpha=0.05}$, green points). The unfilled blue points in the graph are the ones that fulfill a likelihood-ratio criterion that makes them eligible candidates for the next round. This criterion decreases with every round of the algorithm, but is still rather high in this round. The figure therefore contains more points than the points in the joint CR that are counted in the status display on screen.

```
  Status: 14 sets within total CI and 6 within inner. Best fit: 125.8166
Starting round 3, refining a selection of 200 parameter sets, with 40 tries each
```

Round 3, Module 2. Genetic algorithm The algorithm now stays in Module 2 until its targets are met (sufficient points in the joint CI and in the inner rim, unless the maximum number of rounds is reached). The shape of parameter space becomes clearer, and there are more green points.

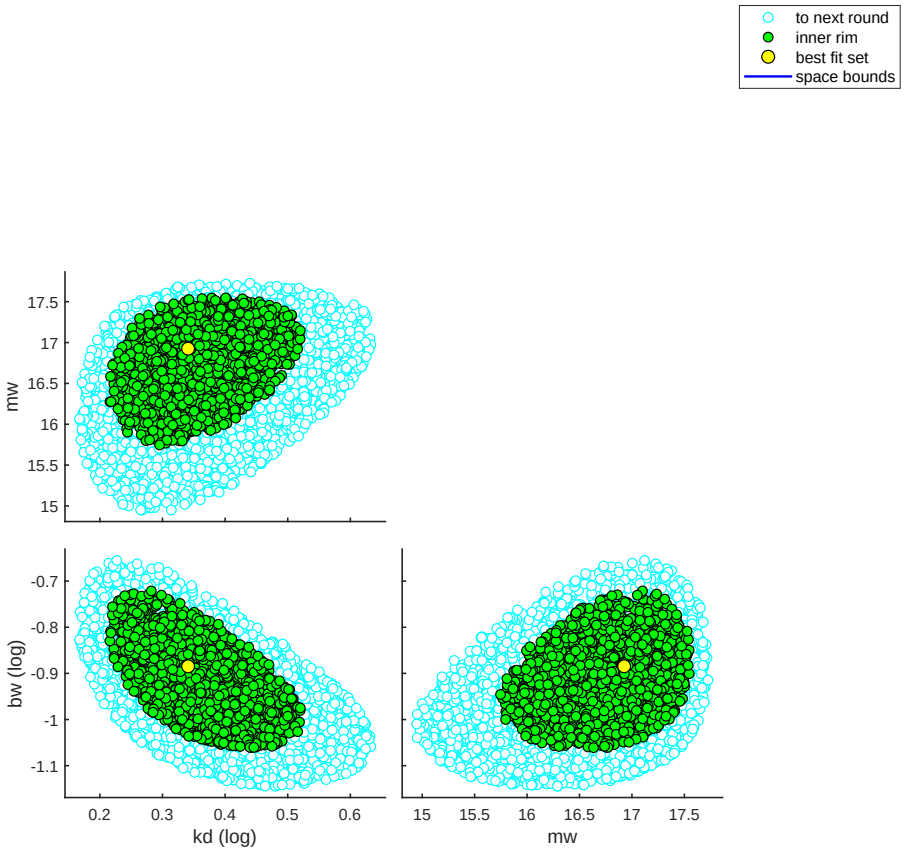


A status update is printed on-screen.

```
  Status: 245 sets within total CI and 86 within inner. Best fit: 125.8154
Starting round 4, refining a selection of 753 parameter sets, with 60 tries each
```

Round 4, Module 2. Genetic algorithm The algorithm now stays in Module 2 until its targets are met. The final shape of parameter space has emerged, and we can now clearly visualise continuous lines of equal likelihood ratio as the outer edges of the blue and green points. These are not the perfect ellipses of Figure 7, which shows that we are not in the 'asymptotic' situation (see Section 2.5). However, in this case, the shapes are not too different from ellipses, which is why this case study can be designated as 'well-behaved'.

Note that the blue points indicate a volume of space that is still somewhat larger than the joint CI of the model parameters.

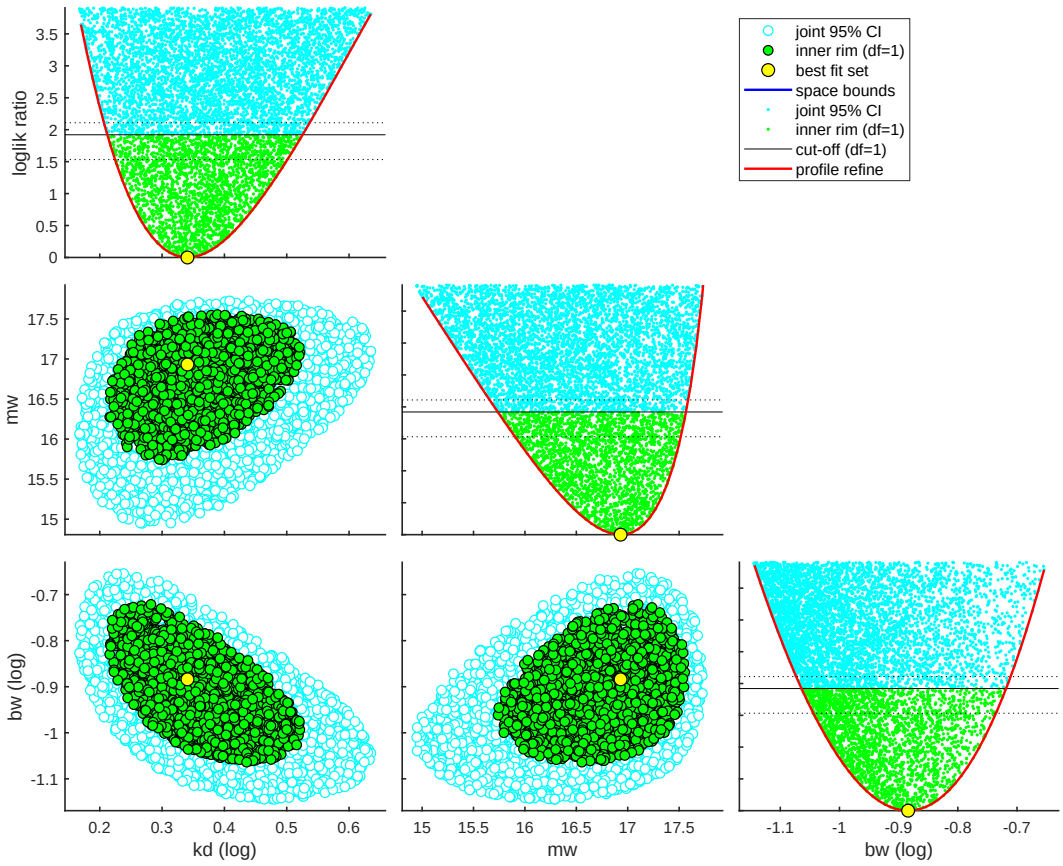


A status update is printed on-screen. At this point, the targets are met: there are sufficient points in the joint CI as well as in the inner rim. A simplex optimisation is ran on the best set so far, which only marginally improves the fit.

```
Status: 7250 sets within total CI and 2407 within inner. Best fit: 125.8154
Finished sampling, running a simplex optimisation ...
Status: 7251 sets within total CI and 2408 within inner. Best fit: 125.8153
```

Round 5, Module 3. Likelihood profiling The algorithm now moves to Module 3. It now also plots the 'side view' of the parameter landscape on the diagonal of the plot. This way, we can see the height of the points (the minus log-likelihood-ratio or MLLR), projected in one of the dimensions of parameter space (such that only one parameter is shown on the x-axis). The green points in the parameter-space plots correspond to the green points in the 'side view' plots on the diagonal. The profile likelihood is the lower edge of the cloud in this 'side view' projection. The algorithm refines this edge by sequential optimisations (red line), which, in this example, is not leading to any surprises.

This plot is equivalent to the schematic plot in Figure 8. However, the profile plots on the diagonal are all 'right side up', as the plot format of Figure 8 does not work so well anymore with more than two parameters. Note the horizontal dotted lines around the critical value of the χ^2-criterion (plotted at $0.5 \times \chi^2_{df=1,\alpha=0.05}$ as the MLLR is plotted on the y-axis). The points within this band will be used for error propagation. This is equivalent to the grey bands in Figure 11.



A status update is printed on-screen. Another simplex optimisation is ran on the best set so far, which only marginally improves the fit (beyond the fourth decimal).

```
Starting round 5, creating the profile likelihoods for each parameter
```

```
Finished profiling, running a simplex optimisation on the best fit set found ...
Status: 7252 sets within total CI and 2409 within inner. Best fit: 125.8153
```

Finishing up. Reporting The algorithm has finished, and no problems were flagged. Therefore, CIs on the model parameters can be calculated (interpolated from the profile likelihood) and printed on screen.

```
=========================================================================
Results of the parameter estimation
=========================================================================
   openGUTS Matlab: v1.0 (10 December 2019)
   Base name      : propiconazole
   Analysis date  : 18-Feb-2020 (09:41)
   Following data sets are loaded for calibration:
     set: 1, file: propiconazole_constant.txt
   Sample: 7252 sets in joint CI and 2409 in inner CI.
   Propagation set: 1114 sets will be used for error propagation.
   Minus log-likelihood has reached 125.82 (AIC=257.63).
Best estimates and 95% CIs on single parameters
=========================================================================
kd    best:      2.191 (     1.630  -     3.353 ) 1/d       fit: 1 (log)
mw    best:      16.93 (     15.74  -     17.57 ) uM        fit: 1 (norm)
hb    best:    0.01307 (       NaN  -       NaN ) 1/d       fit: 0 (norm)
bw    best:     0.1306 (   0.08626  -    0.1912 ) 1/(uM d)  fit: 1 (log)
Fs    best:          1 (       NaN  -       NaN ) [-]       fit: 0 (norm)
=========================================================================
```

Note: the size of the sample has only increased 1 set from the previous round (this is because the result from the simplex optimisation was added to the sample.). However, also the optimised profile points were added to the sample before saving. This is not included in the counters used for reporting on screen, which was an oversight in the openGUTS software v1.0.

The same data set was also used in the GUTS ring test (Appendix A in [10]). The openGUTS test result background document (http://openguts.info/download.html) shows more detailed comparisons between openGUTS and the Bayesian platform MOSAIC/MORSE. The results are very similar, which is to be expected when the inference is dominated by the information in the data set (rather than by the priors).

6 Implementation in BYOM

The parameter-space explorer algorithm was specifically developed for the openGUTS software (see http://www.openguts.info). However, an automated, robust and user-friendly algorithm would also be helpful in other cases, for example also for DEBtox applications. To this end, the openGUTS algorithm was implemented into the general modelling platform BYOM under Matlab (see http://www.debtox.info/byom.html). In principle, the parameter-space explorer can now be used for any model analysis, but a few things need to be kept in mind:

1. The algorithm works well for 1-4 free model parameters (though for 1 free parameter, there is little need for it), and will also generally work (though rather slowly) for 5 free parameters. However, for more parameters, it is quite possible that the algorithm will not provide good coverage of parameter space (and in any case will be extremely slow). One can always restart the algorithm with reduced search ranges, learning from the initial run. Fitting more than 4 parameters is therefore best done by experts until more testing has been performed.

2. The algorithm needs min-max search ranges for each free model parameter. Tighter search ranges will lead to more robust and rapid analyses, but run the risk of missing potentially important parts of parameter space. Selection of search ranges thus requires proper attention. For openGUTS, search ranges are calculated automatically, based on the data set and the foreseen extrapolations (see design document at https://openguts.info/download.html). For DEBtox analyses, a similar set of rules can be used, but this is more complex and needs to be further tested.

3. Linked to the previous point: for each parameter, it needs to be decided whether the parameter is fitted on log-scale or on normal scale. For openGUTS this decision is performed automatically (see design document at https://openguts.info/download.html).

4. The sample from parameter space is saved in a `mat` file. The name is based on the name of the script file from which it is run. The sample can be used to calculate CIs, just like the other methods that are implemented in BYOM (Bayesian and the likelihood-region shooting method).

5. In the openGUTS setting, the algorithm has been optimised (e.g., in terms of min-max bounds) and tested to make sure it can perform robustly without user interference. In other settings, more user expertise is needed.

6.1 Technical notes

Further notes on the implementation in BYOM:

1. The functions needed for the parameter-space explorer are kept in a separate folder `parspace`, as sub-folder of the `engine`. This is helpful to organise the functions

a bit, and also while these functions are distributed under a different license than the rest of BYOM. This separate folder in the `engine` also required modifications to `pathdefine` (which is in every BYOM folder in which scripts are run). The file `pathdefine` needed for the parameter-space explorer is thus *not* the same as the standard BYOM version.

2. In general, when working with BYOM, one would fit a data set first, and calculate CIs on the parameter estimates next. The optimisation routine thus prints the best estimates only. Since the parameter-space explorer does the optimisation and parameter CIs in one go, this does not fit nicely into the BYOM framework. Therefore, the algorithm will produce its own output (with CIs), *before* the standard BYOM output of the optimisation routine.

3. The code for the parameter-space explorer is kept the same as in openGUTS as much as possible, to allow easy updating in the future (e.g., to match any changes to openGUTS in BYOM). Major difference in the structure of the code is that openGUTS collects and follows a standard set of 5 parameters (all of the parameters of the reduced GUTS models), whether they are fitted or not. Since the BYOM implementation needs to be more generic and flexible, it only collects/follows the *fitted* parameters. This has some consequences for the logistics in the code.

4. The adaptation to BYOM was performed to accommodate DEBtox analyses in the near future. Since DEBtox always requires ODE solvers these analyses will be substantially slower than those for GUTS. For this reason, the settings of the algorithm were modified to increase speed, sacrificing some robustness/precision. There is an option `opt_optim.ps_rough` that is by default set to 1 (rough calculations), but can optionally be set to 0 (same settings as openGUTS). Furthermore, an option was included to skip profiling and additional sampling rounds for initial exploration.

5. In general the BYOM framework is distributed under a very open MIT-style license. However, the parameter-space explorer was developed for openGUTS, which is distributed under the GNU GPL V. 3, which is more restrictive. Therefore, the BYOM versions of the openGUTS functions (in the directory `engine\parspace`) are also distributed under the GNU GPL.

7 Detailed analysis of fluorophenyl case study

The main text shows the analysis for GUTS-RED-SD on the data set of fluorophenyl in fathead minnows, performed with the BYOM implementation (https://www.debtox.info/byom.html). Here, the parameter estimates and the CIs are shown as performed with the openGUTS standalone software (https://openguts.info/download.html, v. 1.0), and both the SD and IT models are fitted. The results are compared to the Bayesian inference for GUTS in the MORSE package (version 3.2.5) for R (version 3.6.3) (see https://cran.r-project.org/package=morse), to show the influence of the rules for deriving priors as proposed by [3]. The same rules are used in the on-line automated calculations of MOSAIC (https://mosaic.univ-lyon1.fr/guts, see [1]). Since MOSAIC always fits the background hazard together with the other model parameters, this is here also done for openGUTS.

Many thanks to Benoit Goussen for performing the MORSE calculations in R for me. The MOSAIC software ran into problems with the calculations of the LP10, and does not provide the output for the comparison of priors to marginal posteriors. Therefore, the MORSE calculations are more suitable for the comparison (the fits in MOSAIC were almost identical, as the same engine is used as in MORSE). One modification was necessary: the MFx function in the MORSE package was modified for the SD models in order to use the ODE solver (function `predict_ode`). The calculation returns NAs otherwise.

7.1 GUTS-RED-SD fit

Table 1 below provides the parameter estimates with their confidence/credible intervals. For the openGUTS analysis, this is the best-fitting parameter set with 95% likelihood-based confidence interval. For MORSE, this is the median of the marginal posteriors, with its 0.025-0.975 quantiles. The figures show the most relevant output of both platforms. The fits are here shown in a multipanel figure, which makes them readable when plotting with CIs.

Visually, the two fits are quite similar, and also the optimised parameter estimates are not too different. However, the CIs are very different, especially the lower edges of the CIs for k_d and m_w, and the upper edge of the CI for b_w. For openGUTS, the best value is well defined (as identified from the profiles in Fig. 15), but the CIs include slow kinetics (k_d running away to zero: $-\infty$ on log-scale). The hard min-max bounds prevent the sample from running away, but they do allow very low values of k_d in the CI. The model fit with k_d at its lower boundary of 0.001641 d^{-1} is not significantly worse than the best fit as the profile curve stays below the critical value. Since the profile likelihood for k_d is flat in the lower part of the plot, we can conclude that $k_d = 0$ would also not provide a worse fit (although the model would need to be adapted to represent a k_d of exactly zero, see [10], Appendix C).

Clearly, the MORSE fit does not go into slow kinetics: the lower edge of the CI for k_d is well-defined at 0.23 d^{-1}. Similarly, a large difference with openGUTS is observed for the lower CI of m_w and the upper CI of b_w. The reason lies in the choice of priors, as can be seen in Figure 18. Usually, one would like to see that priors exert little influence

on the posterior, so that the latter is dominated by the information in the data (unless priors can be based on previous experiments). Therefore, the marginal posterior should have a large, narrow peak well within the range of the prior. This is observed for b_w and k_d, but not for h_b and m_w. For the backgound hazard h_b, this is not too interesting: there is no control mortality in this data set, so any low value for h_b is fine (i.e., there will be no difference in the fit if we use $h_b = 10^{-3}$ or $h_b = 10^{-6}$ or even $h_b = 0$). For m_w, the shallow marginal posterior, sitting on the edge of the prior, is of more concern. Even though the prior is unbounded, it assigns a very low *a priori* probability to values of m_w below the lowest exposure concentration in the data set (here 1.8 μM). Therefore, m_w is effectively prevented to go to low values. However, this also blocks k_d from going to low values; slow kinetics can only occur with low values of m_w (and high values for b_w). The reason lies in the fact that GUTS uses *scaled* damage as the property driving toxicity. Low values for k_d imply low values of damage, within the duration of a standard 4-day toxicity test (only in steady state will the scaled damage approach the external concentration). Since there *are* effects observed with very low implied damage levels, m_w must be very low as well. And, unlike h_b, the exact value of m_w now matters considerably. Therefore, the openGUTS parameter-space plot (Fig. 15) shows this tight correlation between m_w and k_d. The MORSE plot (Fig. 17) does not show such strong correlations because a large part of parameter space is effectively cut off. Imagine taking the sample in Figure 15, and cutting out all values for $\log m_w < 0$. Due to the correlations with the other parameters, this also implies cutting out low values for k_d and high values for b_w, and we end up with a comparable result as for the MORSE output.

Looking at the model predictions (LC50 and LP10), the best estimates are reasonably comparable, but the lower edges of the CIs are mostly very different. They are not too different for the 4-d LC50, which is basically an interpolation within the data set. However, for the 28-d LC50, the difference is substantial. For the LP10, matters are worse, especially for the FOCUS profile (which is the longest at 485 days): the openGUTS calculation yields a 60-times higher risk for this chemical (based on the lower edge of the CIs). This is caused by the fact that MORSE penalises slow kinetics with a low *a priori* probability, which will affect long-term extrapolations most.

Table 1: Parameter fits with CIs using openGUTS and the MORSE package for R. Predictions for LC50/LP10 also shown with CIs.

data A SD	openGUTS	MLL = 37.74	MORSE-GUTS	
Param.	Best fit	95% interval	Median	95% interval
k_d	0.2607	0.001641* - 1.036	0.5289	0.2285 - 1.466
m_w	1.536	0.007053 - 2.857	2.278	1.379 - 3.693
h_b	1E-6	1E-6* - 0.01641	3.141E-3	1.574E-4 - 0.02122
b_w	0.2088	0.06646 - 38.24	0.1275	0.05262 - 0.3080
4-d LC50	5.66	4.612 - 7.559	5.97	4.92 - 7.86
28-d LC50	1.743	0.3082 - 3.087	2.54	1.53 - 3.86
LP10 FOCUS	1.72E3	25.8 - 2.76E3	2.36E3	1.58E3 - 3.50E3
LP10 Monitor	2.15E4	1.27E3 - 2.94E4	2.66E4	1.99E4 - 3.36E4

The openGUTS software yielded the following explanation for the asterisk (for the lower edge of the CI of k_d and h_b):

```
* edge of 95% parameter CI has run into a boundary
  (this may also have affected CIs of other parameters)
```

The MOSAIC software yielded the following warnings:

```
- The estimation of the natural instantaneous mortality rate (model
  parameter hb) lies outside the range used to define its prior
  distribution which indicates that this rate is very low and so
  difficult to estimate from this experiment !
- The estimation of Non Effect Concentration threshold (NEC) (model
  parameter z) lies outside the range of tested concentration and
  may be unreliable as the prior distribution on this parameter is
  defined from this range !
```

The latter point is the important one. However, it is not just the estimate for the threshold that is unreliable, owing to the strong correlations between the parameters.

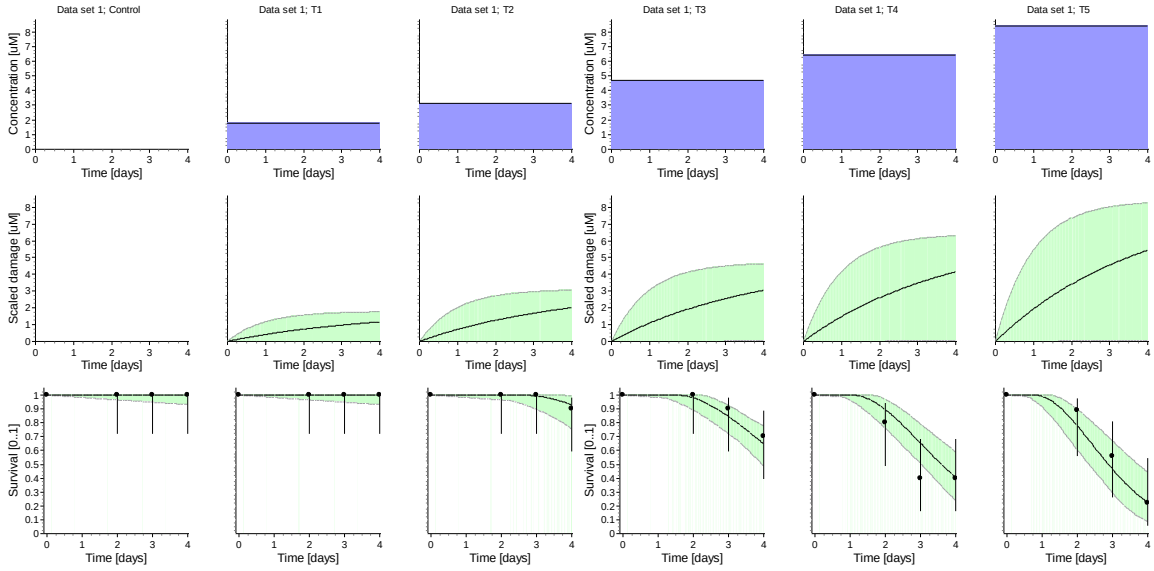


Figure 14: OpenGUTS fit for fluorophenyl SD.

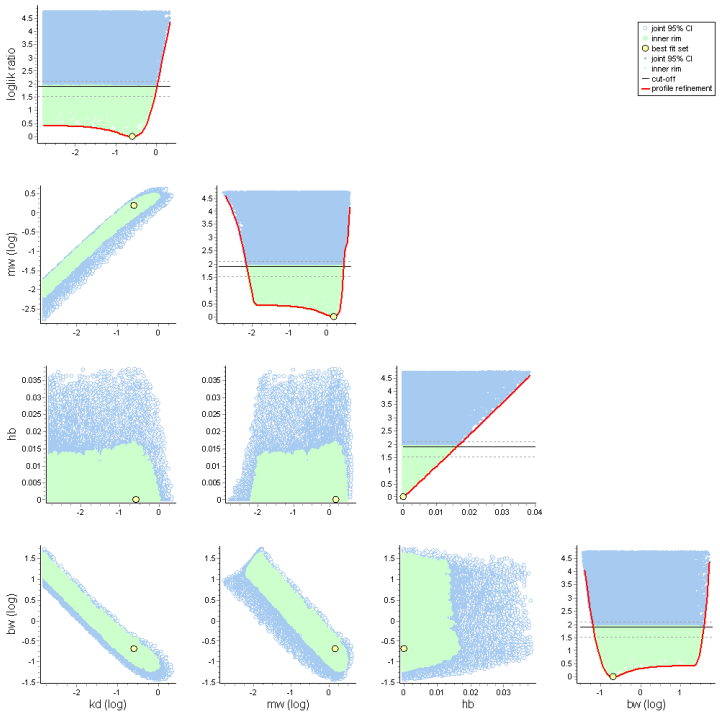


Figure 15: OpenGUTS parameter-space plot for fluorophenyl SD.

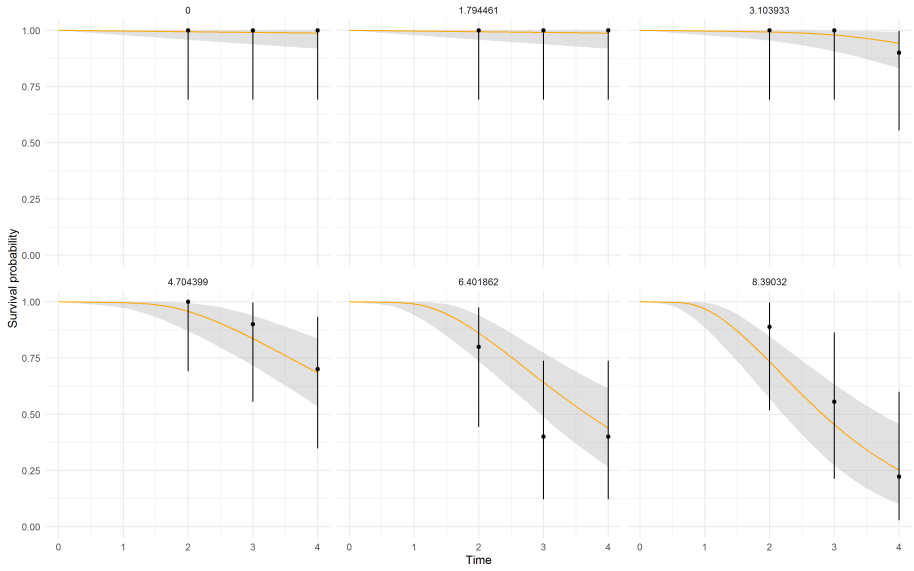


Figure 16: MORSE fit for fluorophenyl SD.

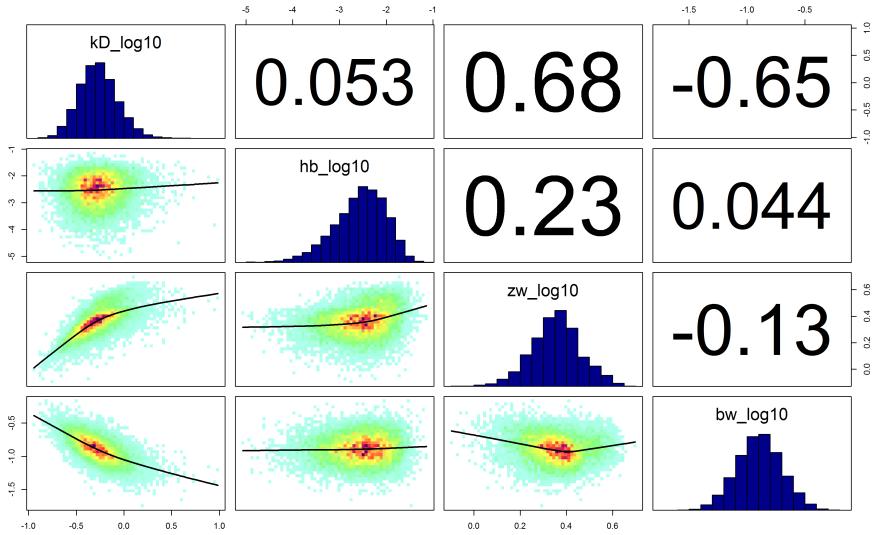


Figure 17: MORSE posterior plots for fluorophenyl SD.

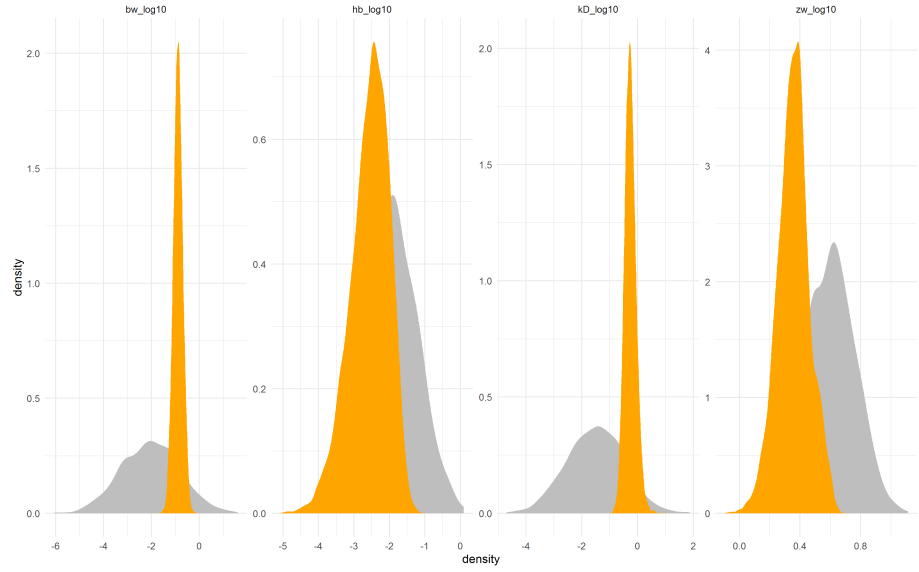


Figure 18: MORSE comparsion priors to marginal posteriors for fluorophenyl SD.

7.2 GUTS-RED-IT fit

Note that openGUTS uses the fraction spread F_s as measure for the width of the threshold distribution, while MOSAIC/MORSE uses the β of the log-logistic distribution. Therefore, openGUTS also recalculates F_s into β, so that is given below as well.

The IT fit also goes into slow kinetics, but in contrast to the SD fit, the best estimate also runs away. Therefore, the best estimate is effectively undefined. This has no consequence for the model fit: at low k_d, the GUTS-RED models reduce from a 4 parameter model to a 3 parameter one. In the IT model, the model behaviour is completely determined by a compound parameter m_w/k_d (for the SD model, another compound parameter will be $b_w \times k_d$). As long as this compound parameter has the same value, the absolute value of the two underlying model parameters becomes irrelevant (and the same fit results). Interestingly, this can be seen from the best estimates of the two platforms: the absolute values are approximately a factor of 100 different, but the ratio m_w/k_d is very similar.

As with the SD fit, the best estimate and the lower edges of the CIs of m_w and k_d cannot go towards very low value as this area of parameter space is penalised by the prior for m_w. This prior assigns low *a priori* probability to m_w values below the lowest tested concentration of 1.8 μM (the lower quantile of m_w's marginal posterior is just below that).

As with the SD fit, the 4-d LC50s are rather comparable, along with their CIs. This is to be expected as the fits (with CIs) are also very similar. However, the extrapolations to longer time scales yield very different results, and now also very different for the best estimate. For the FOCUS profile, the predicted risk is more than 20-times higher in openGUTS than in MORSE (based on the best estimate for the LP10; a factor of 18 on the lower edge of the CI).

Table 2: Parameter fits with CIs using openGUTS and the MORSE package for R. Predictions for LC50/LP10 also shown with CIs.

data A IT	openGUTS	MLL = 38.61	MORSE-GUTS	
Param.	Best fit	95% interval	Median	95% interval
k_d	0.001642	0.001641* - 0.3705	0.1681	0.07256 - 0.3738
m_w	0.0381	0.0319 - 4.92	3.056	1.601 - 5.068
h_b	1E-6	1E-6* - 0.01711	2.684E-3	1.428E-4 - 0.01986
F_s	2.737	1.878 - 4.738		
β	3.638	2.355 - 5.826	3.975	2.404 - 6.064
4-d LC50	5.819	4.844 - 7.468	6.22	5.26 - 7.85
28-d LC50	0.8478	0.7057 - 4.904	3.00	1.62 - 4.96
LP10 FOCUS	77.2	55.8 - 2.96E3	1.80E3	1.02E3 - 2.97E3
LP10 Monitor	3.53E3	2.55E3 - 3.19E4	2.34E4	1.56E4 - 3.28E4

The openGUTS software yielded the following explanation for the asterisk (for the lower edge of the CI of k_d and h_b):

```
* edge of 95% parameter CI has run into a boundary
  (this may also have affected CIs of other parameters)
```

The MOSAIC software yielded the following warnings:

- `The estimation of the natural instantaneous mortality rate (model parameter hb) lies outside the range used to define its prior distribution which indicates that this rate is very low and so difficult to estimate from this experiment !`
- `The estimation of log-logistic median (model parameter alpha) lies outside the range of tested concentration and may be unreliable as the prior distribution on this parameter is defined from this range !`

The latter point is the important one. However, it is again not just the estimate for the median threshold that is unreliable.

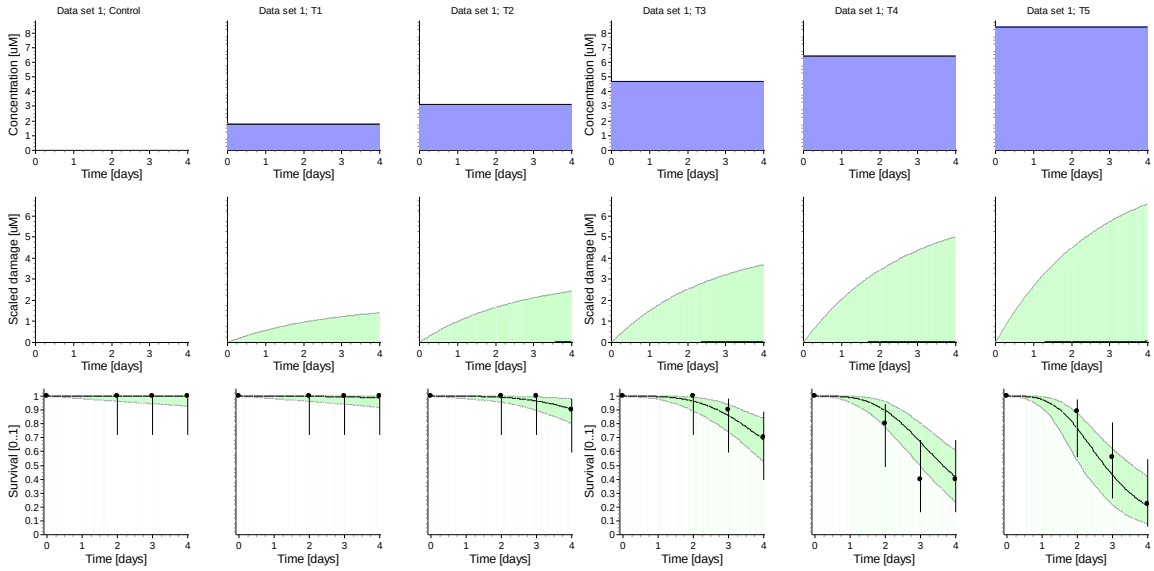


Figure 19: OpenGUTS fit for fluorophenyl IT.

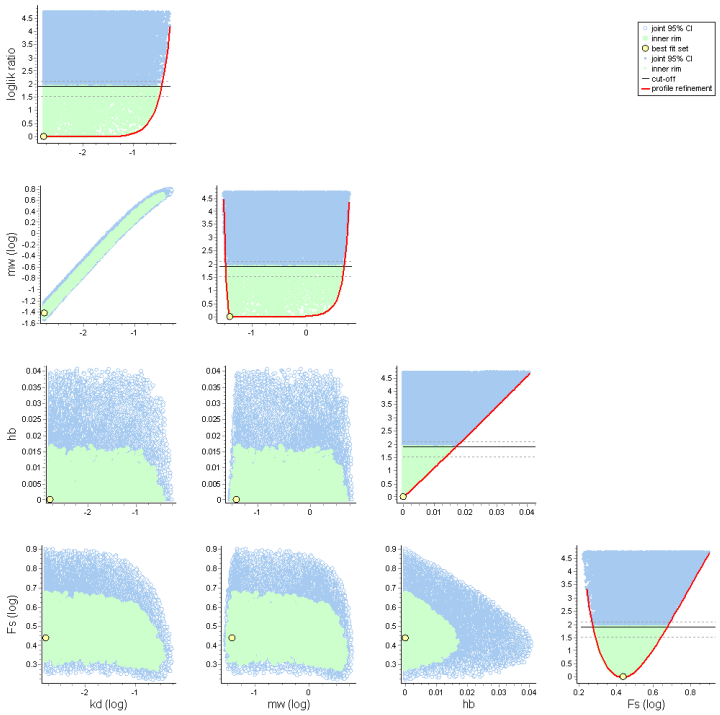


Figure 20: OpenGUTS parameter-space plot for fluorophenyl IT.

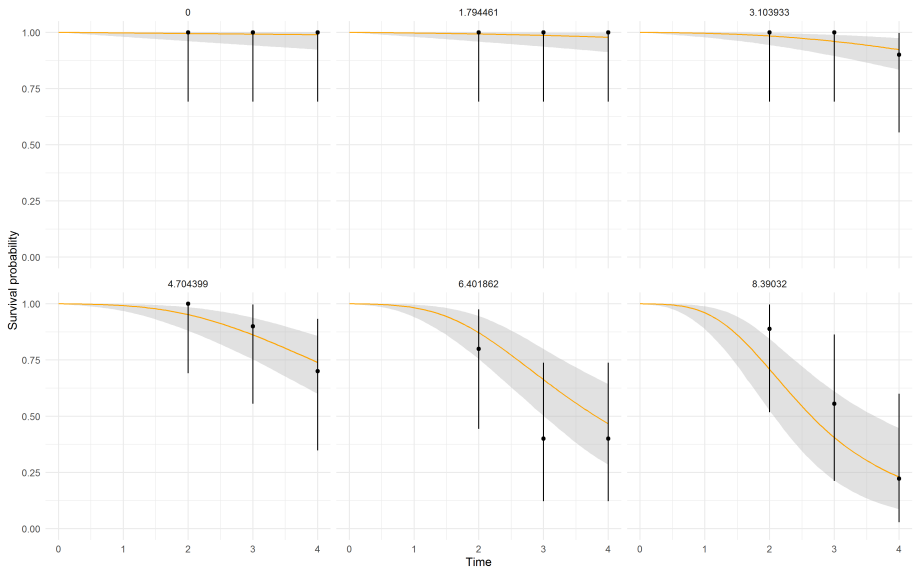


Figure 21: MORSE fit for fluorophenyl IT.

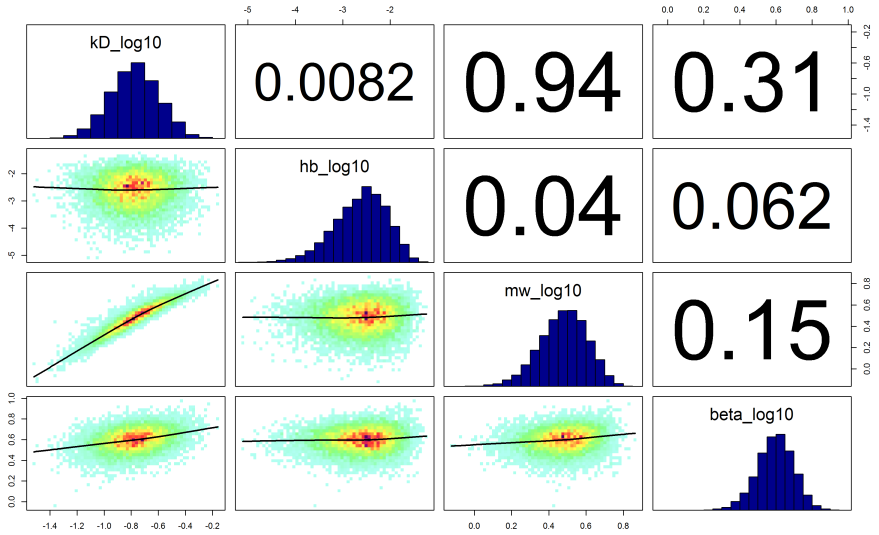


Figure 22: MORSE posterior plots for fluorophenyl IT.

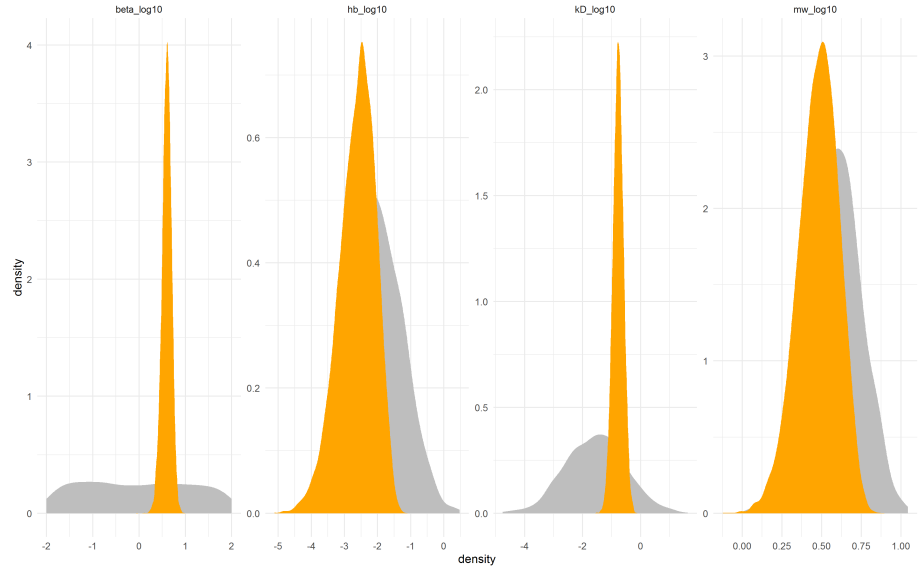


Figure 23: MORSE comparsion priors to marginal posteriors for fluorophenyl IT.

7.3 Conclusions on the comparison openGUTS-MORSE

When the data carry sufficient information on the model parameters, the posterior will be well constrained by the data, the weakly-informative priors of MORSE/MOSAIC do not come into play, and the results will be very similar to those of openGUTS (see test results for openGUTS from the background documents on https://openguts.info/download.html, and ring test results in [10], Appendix A). The results will not be identical as openGUTS presents the best-fitting parameters (the set with the lowest MLL) while MOSAIC/MORSE provides the median of the posterior marginals. Furthermore, the CIs will be somewhat different since they have a different interpretation. Nevertheless, these differences are hardly relevant from the practical standpoint of environmental risk assessment (ERA).

However, when the data set cannot constrain the posterior, the priors begin to exert an influence, and larger differences will ensue. The most problematic case will be 'slow kinetics', which is why such a data set was used for the comparison in this section (in retrospect, such a data set should have been included in the GUTS ring test as well). The results clearly show that the rules for generating priors in MORSE (and MOSAIC, which uses the same rules) prevent the model fit and the model predictions to reflect slow kinetics. The results from a 4-day toxicity test simply cannot contain information on the model parameters when damage dynamics is (or might be) very slow. In the MORSE calculations, this lack of information in the data set is augmented by a specific set of priors, which dominate the posterior and prevent it from going into slow kinetics.

MORSE displays a warning on screen that "The estimation of log-logistic median (model parameter alpha) lies outside the range of tested concentration and may be unreliable as the prior distribution on this parameter is defined from this range !" What this implies is that the threshold m_w wants to be lower than the lowest non-zero test concentration. This is a clear sign of slow kinetics: it is in the structure of the reduced GUTS models that when k_d becomes very low, m_w must also be very low (and for SD, b_w must be high). This is explained in much more detail in Appendix C of the e-book [10]. However, the rule for deriving the MORSE/MOSAIC prior for the threshold m_w is based on the assumption that the threshold should be within the range of tested concentrations. Therefore, a prior is constructed that assigns low *a priori* probability to thresholds below the lowest non-zero test concentration (here: 1.8 μM) [3]. The tables above show that the lower edge of the CI on m_w, as given for MORSE, is indeed not much lower than that value, both for the SD and IT fit. Preventing the posterior to go to low values for m_w also prohibits k_d to go to low values, because of the strong correlation between the parameters (as can be seen in the parameter-space plots from openGUTS). The prior for k_d does not assert that much influence here, since its lower-reasonable bound is set at a value where damage reaches only 0.1% if steady state by the end of the test. For a 4-day toxicity test, this will be at $k_d = 2.5\,10^{-4}$ d^{-1}, which is even lower than the lower bound for k_d as used in openGUTS ($k_d = 1.6\,10^{-3}$ d^{-1}). Therefore, it is the unfortunate choice of prior for m_w that prevents the fit from showing slow kinetics.

It is unclear why this particular rule for m_w's prior distribution was selected, but it was likely a failure to appreciate the model's behaviour under slow kinetics. As a consequence of this specific prior, when the data set allows for slow kinetics, how low k_d and m_w can

go will depend on the, rather arbitrary, choice of the lowest non-zero test concentration in the toxicity test. This will generally have little consequences for the model fit to a 4-day test (also in this case, the model fits are very similar), nor for the 4-d LC50, but it will strongly bias the model predictions for much longer time scales. These predictions will now not reflect the possibility of irreversible effects and thus fail to represent a worst-case interpretation of the data in a regulatory context.

The impact of these subjectively limited priors can be minimised by a more careful choice of their width. Rather than basing these priors on test design only, it would be prudent to also consider the extrapolations foreseen (and to consider the structure of the GUTS models, where low values of k_d imply low values for m_w).

7.4 General remarks on Bayes versus frequentist

As already noted, Bayesian and likelihood-based frequentist model analyses will yield very similar results as long as the priors do not impart information (priors specify our beliefs about the values for the model parameters before looking at the data). When the priors are truly non-informative, the Bayesian analysis will reflect the information in the data set only, which is also true for the frequentist analysis. As an aside, truly objective model analysis is impossible for either framework as subjective choices will go into model building and in the definition of the likelihood function. For GUTS applications, it is a rather common situation that data sets (results from acute toxicity tests) do not carry sufficient information to identify all model parameters. The case study in the section is an example of that, showing 'slow kinetics' ($k_d \to 0$, $m_w \to 0$, $b_w \to \infty$). A similar, but less invasive identifiability problem is that of 'fast kinetics' ($k_d \to \infty$). These and other cases are demonstrated in the 'guide to interpretation' from the openGUTS web page (http://openguts.info/download.html).

In the likelihood-based frequentist approach, identifiability issues are easily recognised (some parameters run into a min-max bound and the likelihood profile becomes flat). As long as the lowest-possible MLL is established, there are no consequences for the defined edges of the CIs (which are based on their MLL relative to the best fit). Therefore, we only need to ensure that the min-max bounds consider the information that can, in principle, be available in the data set (and thus using test design), and the degree of uncertainty that would be relevant for the foreseen extrapolations. For openGUTS that means extrapolation to 485-day FOCUS profiles. A 4-day toxicity test will never contain information on the model parameters for very slow chemicals (low value of k_d). However, such low values are highly relevant from the ERA perspective, when extrapolating to much longer time scales.

For Bayesian analysis, identifiability problems are of greater concern than for likelihood-based frequentist approach, as such issues preclude the use of non-informative priors. When priors are taken as non-informative, and the data do not carry sufficient information, the posterior will be 'improper' (does not integrate to one), causing problems in MCMC analysis and precluding Bayesian inference [4, 18]. The solution is to use priors that *do* carry some information. Few people would object if prior distributions are based on well-established knowledge (e.g., information from previous toxicity tests on the same, or similar, chemical-species combination). However, in general, we do not have such informa-

tion for TKTD models. Automation would thus have to rely on some subjective choices, which needs to be considered carefully.

The choice of priors is a recurring area for debate, also among Bayesians. An example is the discussion of 'objective' [2] versus 'subjective' [5] analysis. There are two points that I like to take away from that discussion. The first is that regulatory authorities will prefer analyses based on objective methodology [2], meaning that priors going into an analysis either do not affect the conclusions, or are based on previous scientific work. The second is that, for a subjective analysis to be widely supported, the priors should reflect the diversity of beliefs held within the scientific community [5], but the "pain to gain ratio" needs to justify such efforts.

To translate these points to the case at hand, automation of Bayesian TKTD analysis in a truly objective sense is out of reach (at least at this moment). Using priors carrying no information may cause Bayesian inference to fail, and there is insufficient knowledge about (patterns in) TKTD model parameters to construct objective informative priors in an automated manner. If subjectively-informed priors are used in ERA, they either need to exert negligible influence on the results (in all cases), or they need to be widely agreed upon. The case study shows the first not to be true for the Bayesian MORSE/MOSAIC tools: the rules for automatic prior selection [3] hamper the analysis to reflect irreversible mechanisms of toxicity. The second statement is also not true: these rules where not discussed with a wider group of ERA stakeholder, and it should be clear that not all experts agree on this particular choice of priors [7]. Until ERA itself is placed into a Bayesian framework, I would conclude on pragmatic grounds that frequentist likelihood-based methods are more suitable in the context of automated TKTD analysis. Identifiability issues can be readily recognised and have limited impacts on the analysis (with a careful choice of min-max bounds), while avoiding a debate on the most appropriate prior distributions.

References

[1] V. Baudrot, P. Veber, G. Gence, and S. Charles. Fit reduced GUTS models online: from theory to practice. *Integrated Environmental Assessment and Management*, 14(5):625–630, 2018.

[2] J. Berger. The case for objective Bayesian analysis. *Bayesian Analysis*, 1(3):385–402, 2006.

[3] M. L. Delignette-Muller, P. Ruiz, and P. Veber. Robust fit of toxicokinetic-toxicodynamic models using prior knowledge contained in the design of survival toxicity tests. *Environmental Science & Technology*, 51:4038–4045, 2017.

[4] A. E. Gelfand and K. Sahu. Identifiability, improper priors, and Gibbs sampling for generalized linear models. *Journal of the American Statistical Association*, 94(445):247–253, 1999.

[5] M. Goldstein. Subjective Bayesian analysis: principles and practice. *Bayesian Analysis*, 1(3):403–420, 2006.

[6] T. Jager. All individuals are not created equal; accounting for interindividual variation in fitting life-history responses to toxicants. *Environmental Science & Technology*, 47:1664–1669, 2013.

[7] T. Jager. Comment on "robust fit of toxicokinetic-toxicodynamic models using prior knowledge contained in the design of survival toxicity tests". *Environmental Science & Technology*, 51(14):8200–8201, 2017.

[8] T. Jager. Revisiting simplified DEBtox models for analysing ecotoxicity data. *Ecological Modelling*, 416:108904, 2020.

[9] T. Jager, C. Albert, T. G. Preuss, and R. Ashauer. General Unified Threshold model of Survival - a toxicokinetic-toxicodynamic framework for ecotoxicology. *Environmental Science & Technology*, 45:2529–2540, 2011.

[10] T. Jager and R. Ashauer. *Modelling survival under chemical stress. A comprehensive guide to the GUTS framework*. Toxicodynamics Ltd., York, UK. Available from Leanpub, https://leanpub.com/guts_book, Version 2.0, 8 December 2018, 2018.

[11] T. Jager and E. I. Zimmer. Simplified Dynamic Energy Budget model for analysing ecotoxicity data. *Ecological Modelling*, 225:74–81, 2012.

[12] C. Kreutz, A. Raue, D. Kaschek, and J. Timmer. Profile likelihood in systems biology. *FEBS Journal*, 280(11):2564–2571, 2013.

[13] C. Kreutz, A. Raue, and J. Timmer. Likelihood based observability analysis and confidence intervals for predictions of dynamic models. *BMC Systems Biology*, 6:120, 2012.

[14] W. Q. Meeker and L. A. Escobar. Teaching about approximate confidence regions based on maximum likelihood estimation. *The American Statistician*, 49(1):48–53, 1995.

[15] A. M. Nyman, K. Schirmer, and R. Ashauer. Toxicokinetic-toxicodynamic modelling of survival of *Gammarus pulex* in multiple pulse exposures to propiconazole: model assumptions, calibration data requirements and predictive power. *Ecotoxicology*, 21(7):1828–1840, 2012.

[16] Y. Pawitan. *In all likelihood: statistical modelling and inference using likelihood*. Oxford University Press, Oxford, UK, 2001.

[17] A. Raue, C. Kreutz, T. Maiwald, J. Bachmann, M. Schilling, U. Klingmüller, and J. Timmer. Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics*, 25(15):1923–1929, 2009.

[18] A. Raue, C. Kreutz, F. J. Theis, and J. Timmer. Joining forces of Bayesian and frequentist methodology: a study for inference in the presence of non-identifiability. *Philosophical Transactions of the Royal Society A-Mathematical Physical and Engineering Sciences*, 371(1984):20110544, 2013.