Dose response curves and other nonlinear curves in Weed Science and Ecotoxicology with the add-on package *drc* in **R**

Christian Ritz & Jens C. Streibig

March 5, 2012

Contents

1 Introduction

This document gives a flavor of how to use the open source programme and environment **R** *http://www.r-project.org/* and, particularly, the add-on package *drc*. **R** is the *lingua Franca* of data analysis and statistics at the LIFE faculty of University of Copenhagen. If this is the first time you are exposed to **R** you can approach it as if you are learning a new language. There is syntax and structure that cannot be violated in a language as in **R**. Only active use of **R** will give you the hang of it. **R** is open source and is free and can be downloaded anywhere and used by anyone. Everybody can look into the engine room of **R** to make sure the calculations are right. With commercial programmes you might get some reasonable manuals, but no access to the engine room of the programme. **R** is like a cookbook with meticulously advices of ingredients and cooking methods. The commercial programmes are more like a menu card; you can choose dishes and see some of the particular ingredients, but not the actual way the dishes are made.

This manual is based on you trying to practice running **R** at the same time as you are reading how to do nonlinear regressions.

In biology there are about four curves that can describe biological and biochemical phenomena. They are shown in Fig. 1. three of them have either two or one asymptotes, e.i. upper and/or lower limits.

The development of the *drc* package began with the dose-response curves commonly used in Weed Science, Pesticide Science and Ecotoxicology, but gradually it has been extended with numerous nonlinear curves commonly used en the life sciences and elsewhere (Fig. 1).

Experience shows that one of the problems with non-linear regressions is the guesses of initial parameters. The engine of the add-on package *drc* is the `drm(y~x..)` function with a self starter, which means that the nuisance of chasing initial parameters is done automatically by the `drm(y~x..)` function.

R works literally by a question-and-answer device: You enter a line with a command and press Enter. Either the programme will do something. If you have written a correct code you might get something useful, otherwise one gets an error message, because you have not obeyed the rules of the language. When **R** is ready for input it displays the prompt (`">"`). Note that **R** distinguishes between capital and small letters.

After having familiarized yourself with the console window, the easiest way to get started with **R** is to develop and store your **R** codes in a standard editor, which comes with **R** and is called *Script*. Other open source editors are available and can be integrated with **R**. Particularly, there is *RStudio* (*URL: http://rstudio.org/download/*)which makes life easier for **R** users. *RStudio* integrates many of the more tedious tasks of reading external data, getting help and save graphics etc. The easiest way to execute *RStudio* after downloading it is just to start it and it will automatically start **R**.

Below there is a collection of examples intended to illustrate how to analyse dose-response curves with the add on package *drc* and another package *nlrwr*, which has some useful facilities and experimental data . The requirements of the student is that she is familiar with some basic statistics and knows how to execute **R** codes.

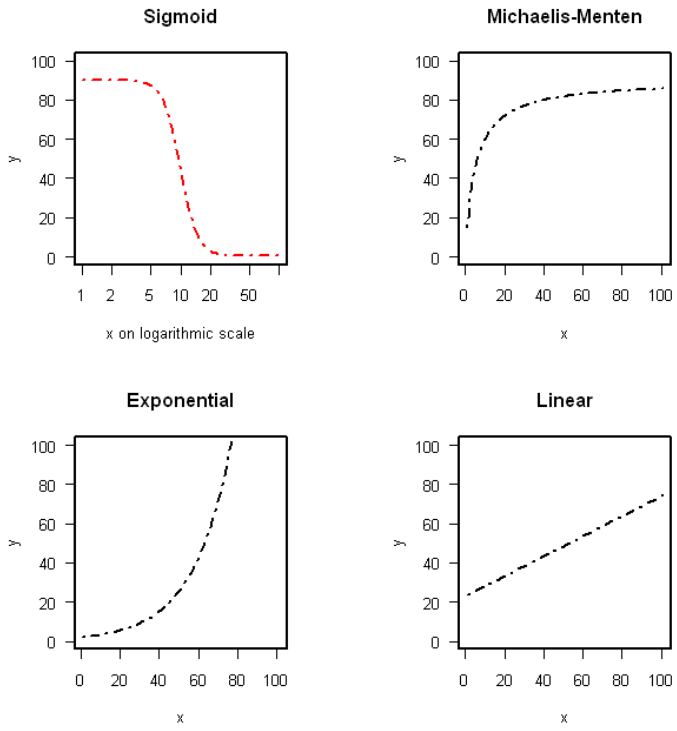

Figure 1: The four curves that can describe most dynamic phenomena in biology and biochemistry. Three of those are nonlinear curves, which have one or sometimes two asymptotes often denoted limits between which the responses range .

The advantage of using regressions analysis in dose-response work instead of the unfortunately popular use of analysis of variance is that you can compare regressions with different choice of doses as long as the doses give a reasonable presentation of the response range. Therefore, you can combine dose-response curves by comparing EDx levels, say ED50, even though the dose ranges are different and the assays have been repeated in time and space with different doses.

2 Install *drc* load it and and getting help

When an **R** session is on, and you have access to the internet you can install the *drc* package by executing the command

```
install.packages("drc", dependencies =TRUE)
```

you will probably be prompted with a selection of a CRAN (Comprehensive R Archive Networks) mirror, just select one close to you and let **R** do the job. The argument `dependencies =TRUE` means that *drc* uses other packages.

Now your are in business and you can begin to work. However, there is one more important detail, you now have the *drc* installed in your **R**, but before working with the facilities in *drc* you must load it by executing:

```
> library(drc)
```

drc is now loaded and you can start to work. It consists of many dataset for exercise purposes. In **R** these datasets are are referred to as dataframe and the first dataframe we will work on is called secalonic. If you wish to see the dataframe you can write

```
> secalonic
```

```
   dose rootl
1 0.000   5.5
2 0.010   5.7
3 0.019   5.4
4 0.038   4.6
5 0.075   3.3
6 0.150   0.7
7 0.300   0.4
```

The dataframe consists of seven observation each with variables called `dose` and the `rootl` . if you wish to know something about the data frame you could write,

```
?secalonic
```

In general you can use this syntax with a question mark and the name of a function or dataframe to get to know what you are working on.

Honestly, the help functionality in **R** is not the easiest to come about. You have to remember that **R** is built by the users (you and me) and and for the users (you and me), including some of the best statisticians, all dedicated and knowledgable people, but they do not put very much effort into help pages. It makes the programme very dynamic but also kind of nihilistic. The next question is: how can I make sure that a certain package does what I want it to do?. This is your responsibility and therefore when you use **R** it is a good idea to explicitly mention which packages you have used. This makes is a problem for the novice to get by. There is, however, help in the numerous contributed documents, you can download in various languages from URL:

`http://cran.r-project.org/other-docs.html`

also there is a dense underwood of commercial books from a variety of publishers that show you how **R** is used in various disciplines.

3 The *drc* engine

The *drc* engine is the function `drm()`, and to run a dose response of the secalonic data you write

```
> drm(rootl ~ dose, data = secalonic, fct = LL.4())

A 'drc' model.

Call:
drm(formula = rootl ~ dose, data = secalonic, fct = LL.4())

Coefficients:
b:(Intercept)  c:(Intercept)  d:(Intercept)  e:(Intercept)
      2.65421        0.09179        5.52975        0.08035
```

Provided you have installed *drc* correctly and loaded it you will see that the execution leaves you with four parameters, `b`, `c`, `d`, `e`. In Fig. 2 the `b` is the relative slope around `e` the ED50, which is the dose required to reduce the `rootl` 50% between the upper limit `d` and lower limit `c`

What has happened is that the `drm` has analyzed the secalonic data (`rool dose`) with a four parameter log-logistic model(`LL.4()`), The argument `data=` points at the dataframe, and `fct=` argument requires a model identification corresponding to one of the several regression models contained in *drc*. You can see some of the regression models *drc* with the usual question mark

```
?LL.4
```

The `LL.4()` indicates that within the parenthesis there are room for more arguments (see later). If you wish to know more about the `drm()` the you just write

```
?drm
```

and you get the possibilities of how to fit curves to data.

If you wish to see how well the regression describes data you can save the results of the regression in an **R** object and get a summary of the regression and a plot the data by running

```
> secalonic.m1<-drm(rootl ~ dose, data = secalonic,
+ fct = LL.4(names=c("Slope","Lower Limit","Upper Limit", "ED50")))
> summary(secalonic.m1)

Model fitted: Log-logistic (ED50 as parameter) (4 parms)

Parameter estimates:

                          Estimate Std. Error    t-value p-value
Slope:(Intercept)        2.6542086  0.6962333  3.8122400  0.0317
Lower Limit:(Intercept)  0.0917852  0.3747246  0.2449404  0.8223
Upper Limit:(Intercept)  5.5297495  0.2010300 27.5070873  0.0001
ED50:(Intercept)         0.0803547  0.0078829 10.1935342  0.0020

Residual standard error:

 0.2957497 (3 degrees of freedom)
```

All relevant regression data are contained in the object `secalonic.m1` and you can see the results by using the `summary` function. The regression parameters now have associated standard error and there is also a test for if the parameters are significantly different from zero. The graphic representation is via the plot function as see below.

The first thing to notice in Fig. 2 is that the dose-axis is on a logarithmic scale, which makes it a bit difficult to picture zero concentration. Therefore, we can use an argument `broken=TRUE` in the `plot` function. The name of the axis are just the name of the variables.

The regression seems to fit to the data pretty well and looking at the summary you notice `e`, which is the ED50, is 0.080 with an associate standard error of 0.008. If your aim was to illustrate the dose-response and use the ED50 to assess the potency of the compound then you have achieve your goal. As you can see in the Fig. 2 the only thing you need to do is to to use the object containing the nonlinear regression as argument in the `plot(secalonic.m1)`.

When you use the command mentions above you evoke a special branch of the general `plot` function designed to plot the mean responses and the fitted response line in *drc*. If you wish to know your possibilities of making graphs more palatable for a publisher you can ask for help

```
?plot.drc
```

```
> plot(secalonic.m1, broken=TRUE)
```

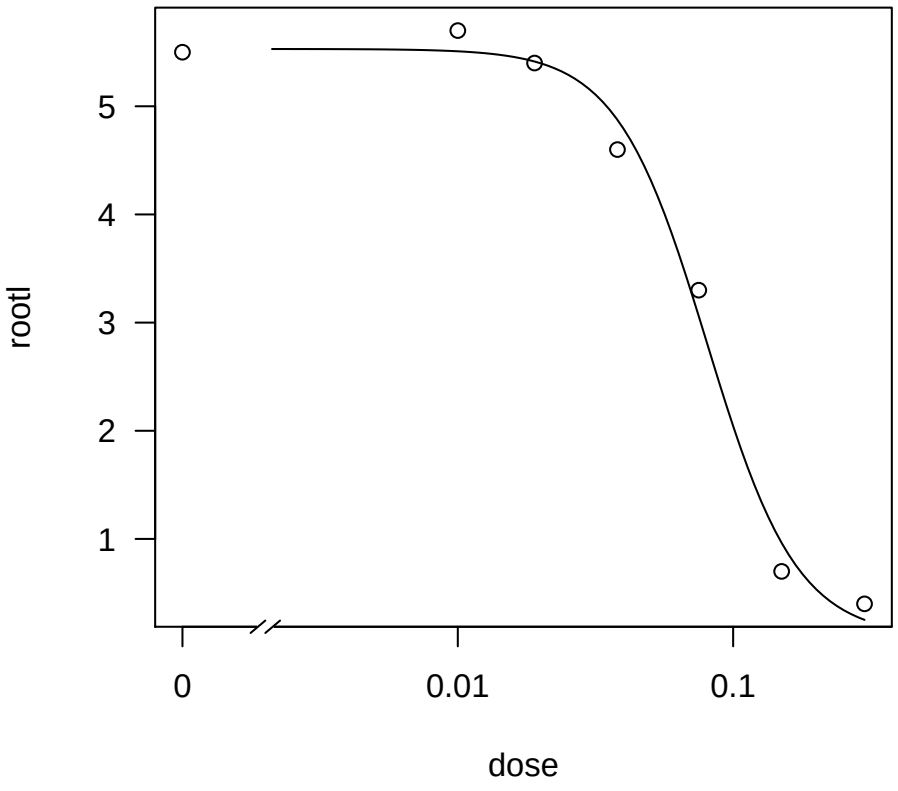

Figure 2: A single fitted dose-response curve plotted together with the original data `secalonic`

Perhaps you wish to see the ED50 with the associated 95% confidence intervals? You just use the function `ED`.

```
> ED(secalonic.m1,50,interval="delta")

Estimated effective doses
(Delta method-based confidence interval(s))

      Estimate Std. Error     Lower  Upper
1:50 0.0803547  0.0078829 0.0552678 0.1054
```

Obviously, we get the ED50 we already know but now also with the lower and upper 95% confidence intervals. If you wish to know more about the `ED` then you do the familiar

```
?ED
```

Although the ED50 is the most used potency estimate for pesticides we often wish to find other EDx levels that are more in accord with the objective of a investigation. For example in ecotoxicology we might be more interested in the ED10 that is the dose required to inflict a 10% effect on an organism. In other instances we wish to find the 90% effect (ED90) to control a pest. You just use the `ED50` function.

```
> ED(secalonic.m1,c(10,50,90),interval="delta")

Estimated effective doses
(Delta method-based confidence interval(s))

      Estimate Std. Error     Lower  Upper
1:10 0.0351149  0.0078689 0.0100726 0.0602
1:50 0.0803547  0.0078829 0.0552678 0.1054
1:90 0.1838789  0.0462775 0.0366034 0.3312
```

The results above in fact satisfy all three end users or regulators of pesticides, the ecotoxicologist gets an ED10, the general toxicologist gets the general accepted potency estimate of ED50, and the farmer gets the effective control level at ED90.

The example with the dataframe `secalonic` is based on mean values taken from a table and since there is not replications, you cannot do very much statistics to substantiate if the model was all right. You can do plots of residuals and see if the distribution is acceptable that is we work with a correct mean function, and the variance homogeneity and normally distributed measuring errors are acceptable. In real life you usually do dose-response studies with replications and then you have even more control over the above prerequisites.

4 Experiments with replication

To carry out a classical dose-response analysis and test for lack-of-fit test of the appropriateness of a chosen model, we need data with replicates. To illustrate this, we fit a dose-response curve to a bioassay with the objective to assess the effect of a natural compound on root growth of a grass species (dataframe `ryegrass`). The variable `conc` is the concentration of ferulic acid in mM and `rootl` is the root length of perennial ryegrass.

```
> head(ryegrass)

     rootl conc
1 7.580000    0
2 8.000000    0
3 8.328571    0
4 7.250000    0
5 7.375000    0
6 7.962500    0
```

you get more information on the data by executing `?ryegrass`. The four-parameter log-logistic model, could be the first choice and note we define the meaning of the parameters by using the `names` argument in `drm`.

```
> ryegrass.m1 <- drm(rootl ~ conc, data = ryegrass,
+ fct = LL.4(names = c("Slope", "Lower Limit", "Upper Limit", "ED50")))
> summary(ryegrass.m1)

Model fitted: Log-logistic (ED50 as parameter) (4 parms)

Parameter estimates:

                       Estimate Std. Error  t-value p-value
Slope:(Intercept)       2.98222    0.46506  6.41251  0.0000
Lower Limit:(Intercept) 0.48141    0.21219  2.26876  0.0345
Upper Limit:(Intercept) 7.79296    0.18857 41.32722  0.0000
ED50:(Intercept)        3.05795    0.18573 16.46440  0.0000

Residual standard error:

 0.5196256 (20 degrees of freedom)
```

As you can see the ED50 is given in the summary of the fit. Sometimes you are not interested in the ED50, but the dose required at the ED90

```
> ED(ryegrass.m1,90,interval="delta")
```

```
Estimated effective doses
(Delta method-based confidence interval(s))

     Estimate Std. Error  Lower  Upper
1:90   6.3886     0.8451 4.6258 8.1515
```

Because we have data with replicates, the function `modelFit()` can be used to obtain a lack-of-fit test, comparing the four-parameter log-logistic to the more general one-way ANOVA model, which has a parameter for each dose level.

```
> modelFit(ryegrass.m1)

Lack-of-fit test

           ModelDf    RSS Df F value p value
ANOVA           17 5.1799
DRC model       20 5.4002  3  0.2411  0.8665
```

The test is far from being significant. It implies that the four-parameter log-logistic dose-response model provides just as good a fit as does the one-way ANOVA model.

The plot in Fig. 3 shows the fit and it seems reasonable. When the well known plot function is used with a drc object then is becomes a modified plot function slightly different for the plot function of the basic **R** as mentioned before. We will not go through all the extra arguments of `plot.drc` function but just whet you interest in exploring the possibilities.

The 4 parameter log-logistic curve use here is a symmetric one around ED50 but perhaps the dose-response curve is not symmetric. We have a list of various curves, e.g. Weilbul1.1 (`W1.4()`) and Weibull.2 (`W2.4()`). You can get to know them by using the familia help by writing `?W1.4`.Figure 4.

First we run the regressions

```
> ryegrass.m0 <- drm(rootl ~ conc, data = ryegrass, fct = LL.4())
> ryegrass.m1 <- drm(rootl ~ conc, data = ryegrass, fct = W1.4())
> ryegrass.m2 <- drm(rootl ~ conc, data = ryegrass, fct = W2.4())
```

and then we contruct the graphs Fig. 4

It is obvious that all three curves yield the same ED50 in Fig. 4, but it is at the high `rootl` or the low `rootl` there are divergence. we can compare by running

```
> ED(ryegrass.m0,c(5,50,95), interval="delta")

Estimated effective doses
(Delta method-based confidence interval(s))
```

```
> plot(ryegrass.m1, broken=TRUE, type="all")
```

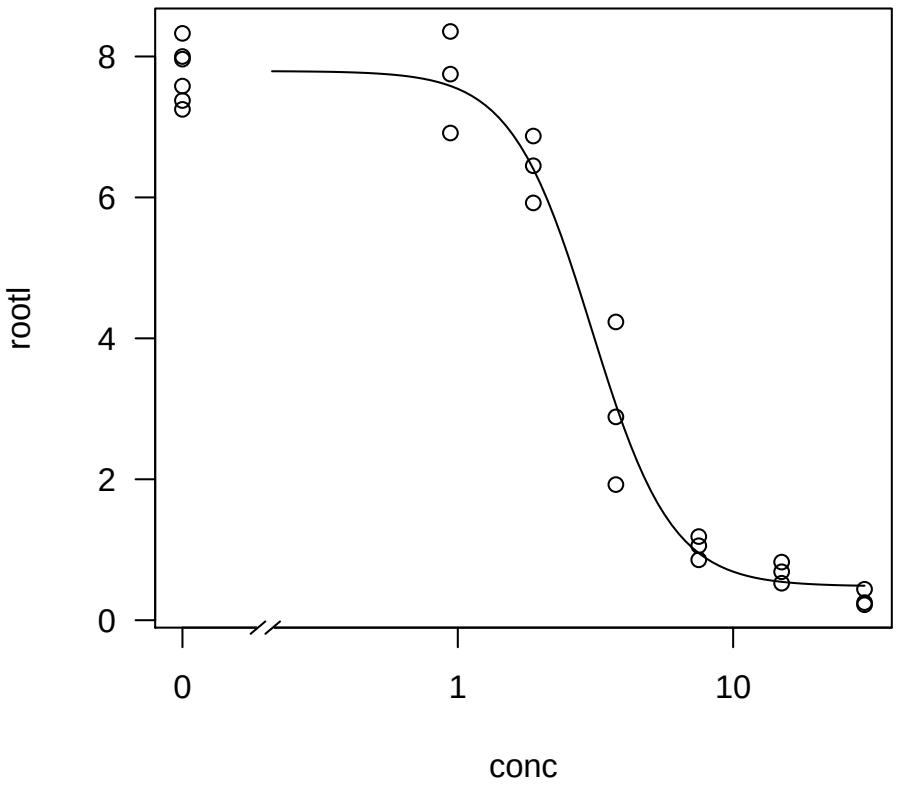

Figure 3: Four-parameter log-logistic curve fitted to the dataframe `ryegrass`.

```
     Estimate Std. Error   Lower   Upper
1:5   1.13930    0.18575 0.75184  1.5268
1:50  3.05795    0.18573 2.67053  3.4454
1:95  8.20774    1.37857 5.33209 11.0834

> ED(ryegrass.m1,c(5,50,95), interval="delta")

Estimated effective doses
(Delta method-based confidence interval(s))

     Estimate Std. Error   Lower  Upper
1:5   1.04079    0.24809 0.52329 1.5583
```

```
> plot(ryegrass.m0, broken=TRUE, xlab="Dose (mM)", ylab="Root length (cm)", lwd=2,
+ cex=1.2, cex.axis=1.2, cex.lab=1.2)
> plot(ryegrass.m1, add=TRUE, broken=TRUE, col="red",lty=2, lwd=2)
> plot(ryegrass.m2, add=TRUE, broken=TRUE, col="blue",lty=3, lwd=2)
> arrows(3, 7.5, 1.4, 7.5, 0.15, lwd=2)
> text(3,7.5, "Weibull-2", pos=4, cex=1.2,col="blue")
> arrows(2.5, 0.9, 5.7, 0.9, 0.15, lwd=2)
> text(3,0.9, "Weibull-1", pos=2, cex=1.2,col="red")
```

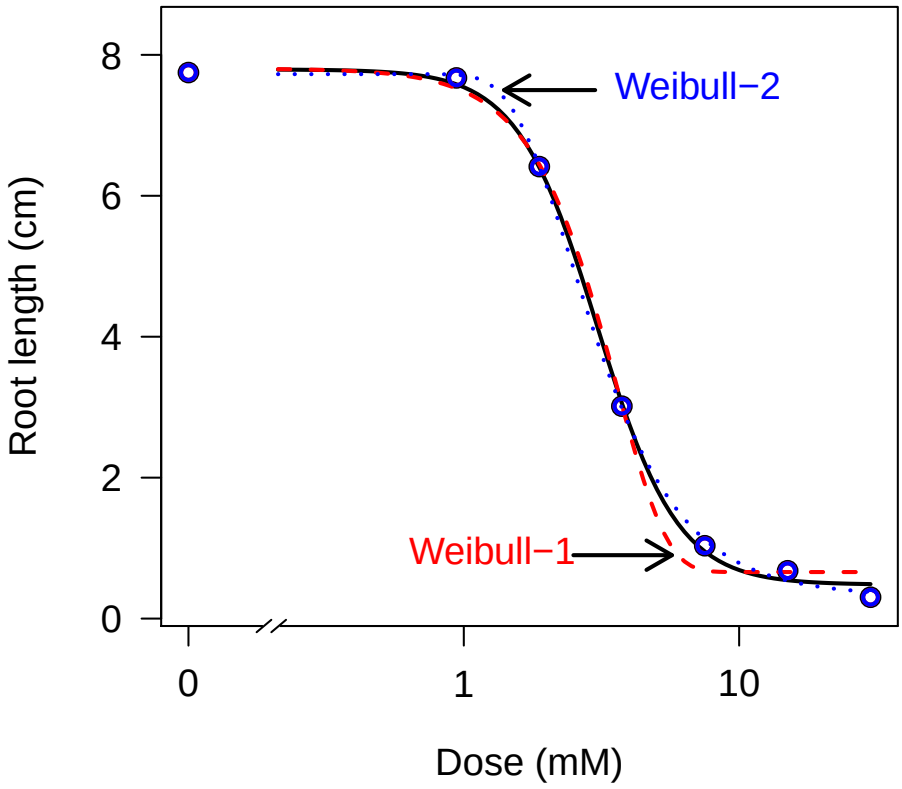

Figure 4: Comparinson of the commonly used three sigmoid dose-response curves,Weilbull.2 has a steep beginning at the upper limit and Weilbul.1 has a steep decrease close to the lower limit `ryegrass`. Note how we can add new curves to the initial plot and annotate the plot.

```
1:50  3.08896    0.17331 2.72744 3.4505
1:95  5.69388    0.68527 4.26443 7.1233
```

```
> ED(ryegrass.m2,c(5,50,95), interval="delta")

Estimated effective doses
(Delta method-based confidence interval(s))

     Estimate Std. Error    Lower   Upper
1:5   1.42447    0.14402  1.12404  1.7249
1:50  2.99691    0.19692  2.58614  3.4077
1:95 11.25317    2.60250  5.82445 16.6819
```

As you see there is not much difference between the curves when it comes to the ED50 values. The various EDx values illustrate the differences in the beginning end and the end of curves. The example above is on the help page for `ryegrass`. It is important to note that the parameter `e` in the log-logistic corresponds to ED50, but this is not the case with the Weilbull functions.

5 Comparing response curves

The results of one dose-response curve cannot stand alone, with or without replications. We often want to compare ED50 levels of several dose-response curves. If we take the two dose-response curves we have fitted so far and want to estimate their relative potency we can do this with by using ED50 of the log-logistic curves for `ryegrass` and `secalonic` with the `comped()` function in *drc*.If you wish to know more about this very versatile function execute `?comped`

```
> comped(c(3.01662 ,0.0803547), c( 0.21005,  0.0078829),
+ log=FALSE, operator = "/")

Estimated ratio of effective doses

     Estimate Std. Error   Lower  Upper
[1,]  37.5413     4.5163 28.6896 46.393
```

where the first vector contains the ED50 parameters and the second one their associate standard errors.

Obviously the secalonic acid in an unidentified species is 37.7 fold more toxic than is the ferulic acid in ryegrass with a 95% confidence interval of 28.6896 to 46.393. Of course the two assays were run under completely different conditions and with different test plants, so the astute reader would ask if it is appropriate to compare them. Often a researcher may have several independent bioassay runs and therefore the `comped` function comes handy into place.

The variabilities of biological assays are huge, if they are not meticulously run under exactly the same conditions. The parameters and their associated standard errors are defined as the probability of getting the results within the 95% confidence intervals of the individual parameters all other things being equal. But unfortunately, all other things are not being equal when dealing with whole organisms and toxicity estimates. Even in test tubes the run on two different days may cause differences of the same magnitude as that of whole organisms. Therefore, when substantiating sensitivity differences in test organisms we have to run the dose-response curves under the same standard conditions, for example by running the bioassays at the same time, or in the laboratory in sequence, but under the same conditions etc.

6 Fitting multiple dose-response curves and model reduction

The data set `S.alba` comprises of 2 dose-response curves. The objective of the experiment was to compare the potency of the two herbicides, glyphosate and bentazon, in *Sinapis alba*. You can find more information on the dataframe by writing `?S.alba`.

```
> head(S.alba)
```

```
  Dose  Herbicide DryMatter
1    0 Glyphosate       4.7
2    0 Glyphosate       4.6
3    0 Glyphosate       4.1
4    0 Glyphosate       4.4
5    0 Glyphosate       3.2
6    0 Glyphosate       3.0
```

The response is `DryMatter` on the `Dose` and the separation of the two curves is `Herbicide`. Often when you have some experiments with more than one response curve it is a good idea to see the individual results for each `Herbicides`. In **R** there is a very versatile function called `xyplot(..)` as seen in Fig. 5. Note that you need to load the library *lattice* that comes with the basic **R**.

We do a simultaneous model fit assuming different individual parameters for each of the two curves.

```
> S.alba.m1<-drm(DryMatter~Dose, Herbicide, fct=LL.4(), data=S.alba)
> summary(S.alba.m1)
```

```
Model fitted: Log-logistic (ED50 as parameter) (4 parms)

Parameter estimates:

              Estimate Std. Error   t-value p-value
b:Glyphosate  2.715409   0.748279  3.628873   6e-04
b:Bentazone   5.134810   1.130949  4.540266   0e+00
c:Glyphosate  0.891238   0.194703  4.577429   0e+00
c:Bentazone   0.681845   0.095111  7.168925   0e+00
d:Glyphosate  3.875759   0.107463 36.066087   0e+00
d:Bentazone   3.805791   0.110341 34.491101   0e+00
e:Glyphosate 62.087606   6.611444  9.390929   0e+00
e:Bentazone  29.268444   2.237090 13.083268   0e+00

Residual standard error:

 0.3730047 (60 degrees of freedom)
```

```
> library(lattice)
> xyplot(DryMatter~log(Dose+.1)|Herbicide, data=S.alba)
```

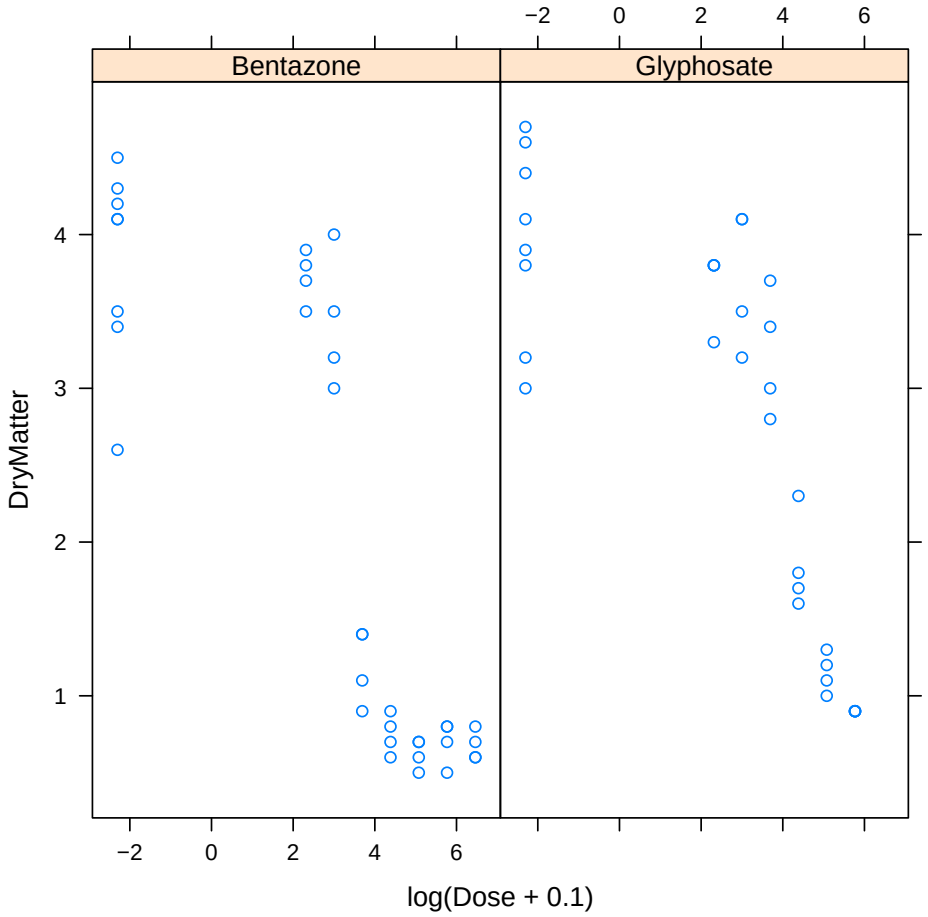

Figure 5: A plot with the two curves. Note that within the function `xyplot` you can define a logarithmic x-axis by `log(Dose+1)`. Information of data you get by executing `?S.alba`, and information on xyplot you can get by writing `?xyplot`

```
> modelFit(S.alba.m1)

Lack-of-fit test

          ModelDf    RSS Df F value p value
ANOVA          53 8.0800
DRC model      60 8.3479  7  0.2511  0.9696
```

We notice that the parameters for the upper and lower limits were almost the same for the two herbicides and that the test for lack of fit was nonsignificant, meaning that the regression is just as good to describe the data as is an ordinary ANOVA. We could try to test if the two curves have the same upper and lower limit. From a biological point of view it would make it much easier to give a succinct summary of the experiment. In order to do that we need to use an argument in the `drm` function

`pmodels=data.frame(Herbicide,1,1,Herbicide)`

Since the classification of the two curves are based upon `Herbicide` we can define which parameters the curves should have in common. In the argument above the `data.frame(..)` determines which parameters are to be similar for each curve, and the sequence is in alphabetic order, `b,c,d,e` for the four parameter log-logistic model. Therefore, the argument above makes the `c` and `d` parameter common for the two curves

```
> S.alba.m2<-drm(DryMatter~Dose, Herbicide, fct=LL.4(), data=S.alba,
+ pmodels=data.frame(Herbicide,1,1,Herbicide))
> summary(S.alba.m2)

Model fitted: Log-logistic (ED50 as parameter) (4 parms)

Parameter estimates:

               Estimate Std. Error   t-value p-value
b:Bentazone    5.046141   1.040135  4.851430       0
b:Glyphosate   2.390218   0.495959  4.819387       0
c:(Intercept)  0.716559   0.089245  8.029117       0
d:(Intercept)  3.854861   0.076255 50.551925       0
e:Bentazone   28.632355   2.038098 14.048566       0
e:Glyphosate  66.890545   5.968819 11.206663       0

Residual standard error:

 0.3705151 (62 degrees of freedom)
```

<indent and now we can make a test for lack of fit between the first regressions fit without any restriction upon anyone parameter and one with similar upper and lower limits for both curves.

```
> anova(S.alba.m1,S.alba.m2)

1st model
 fct:     LL.4()
 pmodels: Herbicide, 1, 1, Herbicide
2nd model
 fct:     LL.4()
 pmodels: Herbicide (for all parameters)

ANOVA table

          ModelDf    RSS Df F value p value
2nd model      62 8.5114
1st model      60 8.3479  2  0.5876  0.5588
```

Apparently we can tentatively assume that the reduction in the number of parameters were all right, because the F-test for eh model reduction was not significant. It is noted that there were only minor differences in the regression parameters.

On the basis of the fit above and illustrated in Fig. 6; we can now find the relative potency at the ED50 level by the function `SI()`, which stands for Selectivity Index.

```
> SI(S.alba.m2, c(50,50))

Estimated ratios of effect doses

                            Estimate Std. Error    t-value p-value
Bentazone/Glyphosate:50/50   0.428048   0.043915 -13.024074       0
```

It is obvious above that the `SI` gives the relative potency at the ED50 level with bentazon in the denominator and glyphosate in the numerator and their associate standard errors. For various reasons, however, we would like to present the reciprocal result. This is done by using the additional argument `reverse=TRUE`

```
> SI(S.alba.m2, c(50,50),reverse=TRUE)

Estimated ratios of effect doses

                           Estimate Std. Error t-value p-value
Glyphosate/Bentazone:50/50  2.33619    0.23968 5.57493       0
```

Whatever the reference herbicide the results is the same the relative potency between the two herbicides is significantly different from 1.00, which is the test with the `SI` function. The advantage of a model with common lower limits (c) and upper limits (d) is that

```
> plot(S.alba.m2,broken=TRUE)
> plot(S.alba.m1,broken=TRUE,col="red",lty=c(3,4),add=TRUE)
```

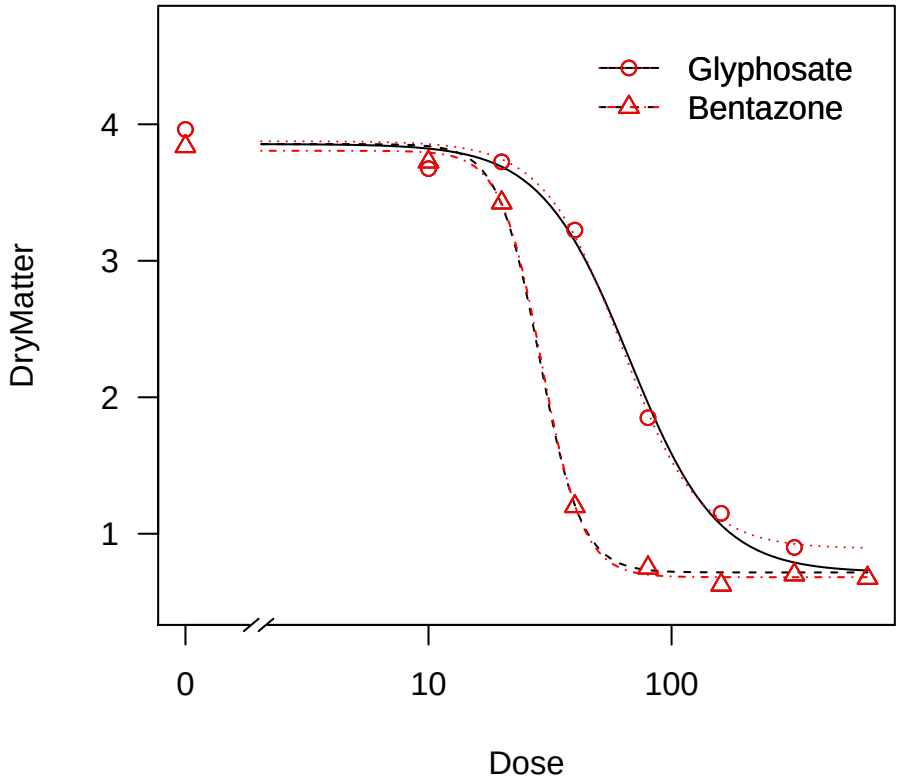

Figure 6: Regression of `S.alba` with identical upper and lower limits compared to the red curves with no restriction on the parameters.

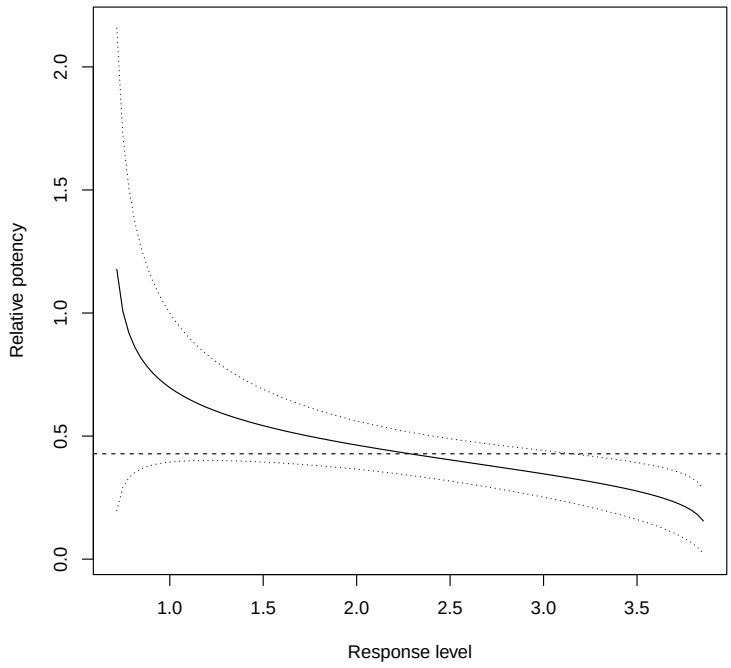

Figure 7: Plot of the the relative potency in response to the response levels with the 95% confidence intervals, the horizontal line shows the relative potency at the ED50.

it renders the relative potency between the two herbicides have the same frame of reference by using the common upper and lower limits.

Figure 6 show that the relative potency at the ED50 level is different form other ED-levels; for example if you wish to use the ED90.

Therefore, when defining a relative potency when the dose-response curves have different slopes it is imperative that the ED-level is given. If the two curves in Fig. 6 are identical in all parameters except the ED50 then we say that the two curves are identical. Is means that the two curves are similar and that the only difference between them is caused by the horizontal displacement along the log(x) axis by different ED50. If this is true then we only need to report one relative potency because it will be independent of the response level be it ED50 or ED90. In *drc* there is a function `relpot` that illustrate the point in Fig. 7.

```
> relpot(S.alba.m2, interval = "delta")
```

It is obvious that the selectivity index definitely depends upon the ED-level and the further you go to either of the limits the more extreme the selectivity index becomes.

Selectivity indices are used by chemical companies to quantify selectivity in their research and development of new herbicides. Since herbicides have an effect on any plant, be

it crops or weeds, we sometimes have to accept a small decrease in the crop yield, for example a 10% decrease might be tolerated (ED10), while a 90% control of the weed (ED90) is considered a reasonable control level. The larger the ED10 for the crop divided by ED90 for the weed become the more selective is a herbicide. The same applies to formulation of hypotheses in ecotoxicology where we wish to compare the relative sensitivity of organisms to various xenobiotics. Which response level should we use: ED10, ED25 or ED50, and do statistical procedures warrant a low ED10 protection level considering the variation in the data?

For the `S.alba` the comparison of the ED50 or any other EDx level is straight forward because after a bit of parameter reduction we could safely assume that the upper and lower limit of the two curves were the same. The ED50 is the dose that half the response between the upper and/or lower limit, but what if we have large differences between the upper and lower limits? An experiment illustrating the effect of two formulation of a contact herbicide on *Galium aparine* shows the problem.

```
> galium.m1 <- drm(drymatter ~ dose, treatment, data = G.aparine, fct = LL.4())

Control measurements detected for level: 0

> summary(galium.m1)

Model fitted: Log-logistic (ED50 as parameter) (4 parms)

Parameter estimates:

      Estimate Std. Error    t-value p-value
b:1    1.81268    0.37821    4.79272       0
b:2    1.39772    0.19740    7.08071       0
c:1  513.33517   21.77275   23.57696       0
c:2  132.29916   30.72047    4.30655       0
d:1  960.89546   14.17267   67.79919       0
d:2 1057.78381   30.26708   34.94833       0
e:1   54.72989    8.30812    6.58752       0
e:2   82.61896    8.46976    9.75458       0

Residual standard error:

 109.251 (232 degrees of freedom)

> modelFit(galium.m1)

Lack-of-fit test

           ModelDf     RSS Df F value p value
ANOVA          219 2601788
DRC model      232 2769103 13  1.0833  0.3747
```

As shown in the summary for the parameters and the graphs in Fig. 8 and test for lack of fit is not significant, so we can entertain the idea that the regressions are just as good to describe the data as is an ANOVA. It is noted that the ED50 must represent different absolute biomass, because the ED50 is defined as the dose yielding a 50% reduction between the upper and lower limits, the latter being grossly different for the two curves. Still we can calculate the relative potency

```
> SI(galium.m1, c(50,50),interval="delta")

Estimated ratios of effect doses
(Delta method-based confidence interval(s))

          Estimate   Lower  Upper
1/2:50/50  0.66244 0.42336 0.9015
```

which is different from 1. But the dry matter for preparation no 1 at ED50 is 736 and for preparation 2 it is 598, and below a dry matter of about 513 you cannot compare the curves at all, because the high lower limit for preparation no. 1. In Fig 8 you can see that the absolute response levels are different form the two curves.

However, we would like to see how the relative potency changes at various response levels by using the function `relplot()`

In Fig. 9 we see how the relative potency varies at different ED levels and that it is only defined until the biomass is 513. The problem here does not go away if the responses are scaled to say percent of the untreated control, a widely used practice. Perhaps it is better to define an absolute comparison, for example at the ED50 of the first curve, which is 737 drymatter ((513.3+960.9)/2)

```
> SI(galium.m1, c(737,737), type="absolute", interval="delta")

Estimated ratios of effect doses
(Delta method-based confidence interval(s))

            Estimate   Lower  Upper
1/2:737/737  1.04322 0.63789 1.4485
```

and now the relative potency is not different from 1.0 (See also Fig. 8). This dataset is a good example of how difficult it becomes to compare potency between curves at a certain EDx when the upper and lower limits are very different among curves. At the end of the day the choice is on the discretion of the biologist not the statisticians.

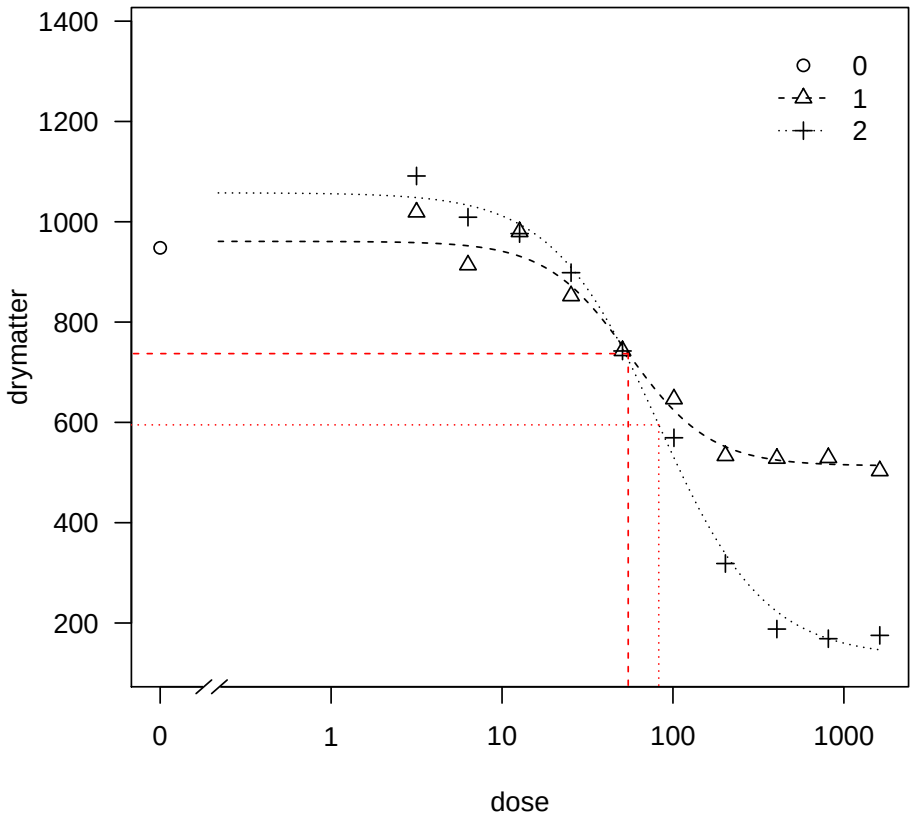

Figure 8: Plot and regression fit of the dataframe `G.aparine` showing the ED50 of the two curves. The absolute responses for ED50s vary from the two curves

```
> relpot(galium.m1, interval = "delta")
```

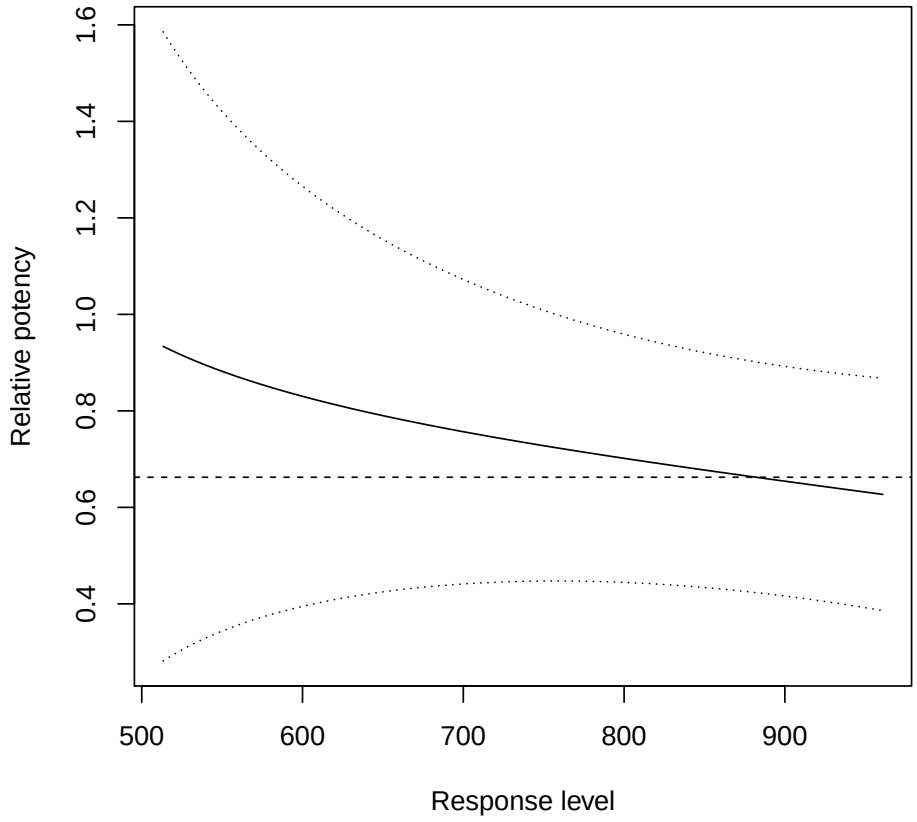

Figure 9: Plot of the the relative potency in response to the response levels with the 95% confidence intervals and the horizontal line shows the relative potency at the ED50.

6.1 Taking charge of the upper and lower limits

The dose-response curves above all had a rather nice distribution of responses, so the estimation of the upper and lower limits is not a problem. However, in some instances it could be tested if the lower limit is necessary. If the lower limit is not different from zero as is the case for secalonic (page 6 you can omit it and run a `LL.3()` model).

However, test statistics does not do it all; you have to use your biological knowledge to determine if the upper limits are reasonable also . The upper limit should be rather close to the untreated control and if you work with growth related responses the lower limit must be positive. If the upper limit is way out of the untreated control there is something wrong and you either have to do the experiment once more or fix the upper limit at a biologically reasonable value. If the lower limit is below zero or unrealistic in any way you have do do the same as for the upper limit. In this particular instance it is you biological knowledge of the system that counts not fancy statistical tricks.

The result of he fitting in Fig. 2 shows that the lower limit was not significantly different form zero. Running the Secalonic once more with `fct=LL.3()` will show you what happens. Try to do it yourself, in fact very little happens so perhaps it was not worth the efforts?

In some instances we have to fit dose-response curves to subjective measurement such as visual effect rating. Per definition they vary between 0 to 100

An example is from an experiment with rating of effects of diquat on *Vulpia bromoides*. The range of effects was theoretical between 0 to 100.

First we have to read the data. by using the `read.csv2()` functions. If you wish to know more about this useful function to read you own data into the system, execute `?read.csv2`.

```
> Vulpia.bromoides<-read.csv2(file.choose())
>  head(Vulpia.bromoides)

  Dose Effect
1    0      0
2    0      0
3    0      0
4   87      0
5   87      0
6   87      0
```

and look at the first 6 observations

```
>  vulpia.bromoides.m1<-drm(Effect~Dose, fct=LL.3(),
+                     data=Vulpia.bromoides)
> summary(vulpia.bromoides.m1)

Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 (3 parms)
```

```
Parameter estimates:

                 Estimate  Std. Error     t-value p-value
b:(Intercept)   -0.880350    0.087938  -10.011071  0.0000
d:(Intercept)  143.994904   25.760427    5.589772  0.0000
e:(Intercept) 3995.005085 1650.601667    2.420333  0.0196

Residual standard error:

 4.789312 (45 degrees of freedom)
```

```
> modelFit(vulpia.bromoides.m1)

Lack-of-fit test

          ModelDf     RSS Df F value p value
ANOVA          40  800.83
DRC model      45 1032.19  5  2.3111  0.0618
```

We run a `LL.3()` and get the results above. The test for lack of fit is on the borderline, but from a biological point of view, the upper limit `d`, is way above the maximum of 100% and the ED50 is with a very large standard error. By looking at Fig. 10 we know why. It is a good example of model fitting and associate statistics that make no biological sense because the upper limit is ridiculously high compared to the actual one.

In fact the large upper limit reflects that we do not have enough observation at the upper end of the curve (Fig 10) and consequently, we have to take charge of the upper limit by defining it a priori using the argument `fixed=c(NA,100,NA)`. If you recall the parameters are sorted alphabetically `a, b, c`, so the first `NA` means that `b` can vary freely, `d` is fixed at 100 and the `NA` for `e` means that ED50 can vary freely.

```
> vulpia.bromoides.m2<-drm(Effect~Dose,
+                          fct=LL.3(fixed=c(NA,100,NA)), data=Vulpia.bromoides)
> summary(vulpia.bromoides.m2)

Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 (2 parms)

Parameter estimates:

                 Estimate  Std. Error     t-value p-value
b:(Intercept)   -1.127601    0.055313  -20.385689       0
e:(Intercept) 1777.795974   75.248397   23.625699       0

Residual standard error:

 5.206975 (46 degrees of freedom)
```

```
> plot(vulpia.bromoides.m1,broken=TRUE, xlim=c(0,10000))
```

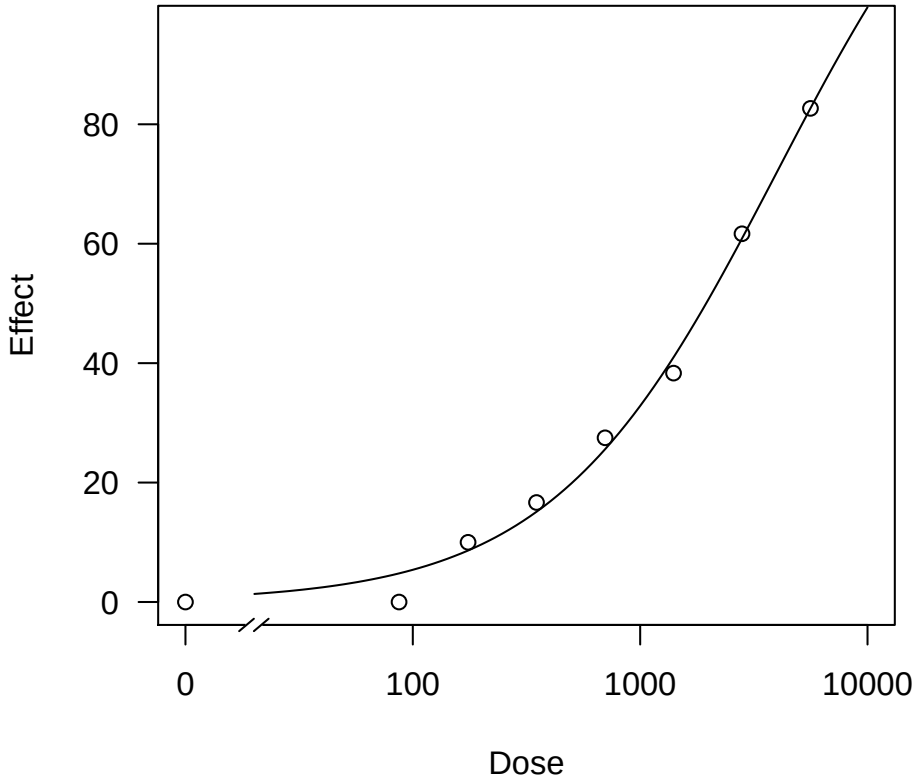

Figure 10: Regression fit of *vulpia* without restriction on the upper limits.

```
> modelFit(vulpia.bromoides.m2)

Lack-of-fit test

          ModelDf     RSS Df F value p value
ANOVA          40  800.83
DRC model      46 1247.18  6  3.7157  0.0050
```

Now the upper limit is fixed at 100% the ED50 has dropped considerably with a small standard error, but the test for lack of fit is now highly significant and inspecting of Fig. 11 shows us why, the mean observation are now systematically diverging form the regression lines. The problem here is that the curve does not look symmetric and therefore we could try with one of the `Weibull` function, but we have to calculate the ED separately because the `e` parameter in the `Weilbull` function is not the same as ED50 such as in the log-logistic model.

```
> vulpia.bromoides.m3<-drm(Effect~Dose,
+                          fct=W2.3(fixed=c(NA,100,NA)), data=Vulpia.bromoides)
> summary(vulpia.bromoides.m3)

Model fitted: Weibull (type 2) with lower limit at 0 (2 parms)

Parameter estimates:

                Estimate Std. Error    t-value p-value
b:(Intercept) 8.5285e-01 3.5133e-02 2.4275e+01       0
e:(Intercept) 2.9651e+03 1.1618e+02 2.5521e+01       0

Residual standard error:

 4.762427 (46 degrees of freedom)

> modelFit(vulpia.bromoides.m3)

Lack-of-fit test

          ModelDf     RSS Df F value p value
ANOVA          40  800.83
DRC model      46 1043.31  6  2.0186  0.0857

> ED(vulpia.bromoides.m3,50,interval="delta")

Estimated effective doses
(Delta method-based confidence interval(s))

     Estimate Std. Error    Lower Upper
1:50 1929.334     69.403 1789.634  2069
```

And now the test for lack of fit is not significant. So this last `Weibull` fit seems to satisfy both the statistical test for lack of fit and the biological rationale that an effect cannot exceed that of 100%. The problem here is that the curve is not symmetric and therefore we could try with one of the function, In Fig. 11 you can see that the dose-response curves look as if there is not much difference between them, but there is if you extend the plots.

```
>  plot(vulpia.bromoides.m2,broken=TRUE,ylim=c(0,150),
+ xlim=c(0,100000),lwd=2,col="blue", lty=3)
>  plot(vulpia.bromoides.m1, xlim=c(0,100000),add=TRUE,
+ pch=21,lty=2,col="red", lwd=2)
>  plot(vulpia.bromoides.m3, xlim=c(0,100000),add=TRUE,
+ pch=21,lty=1,col="black", lwd=2)
```

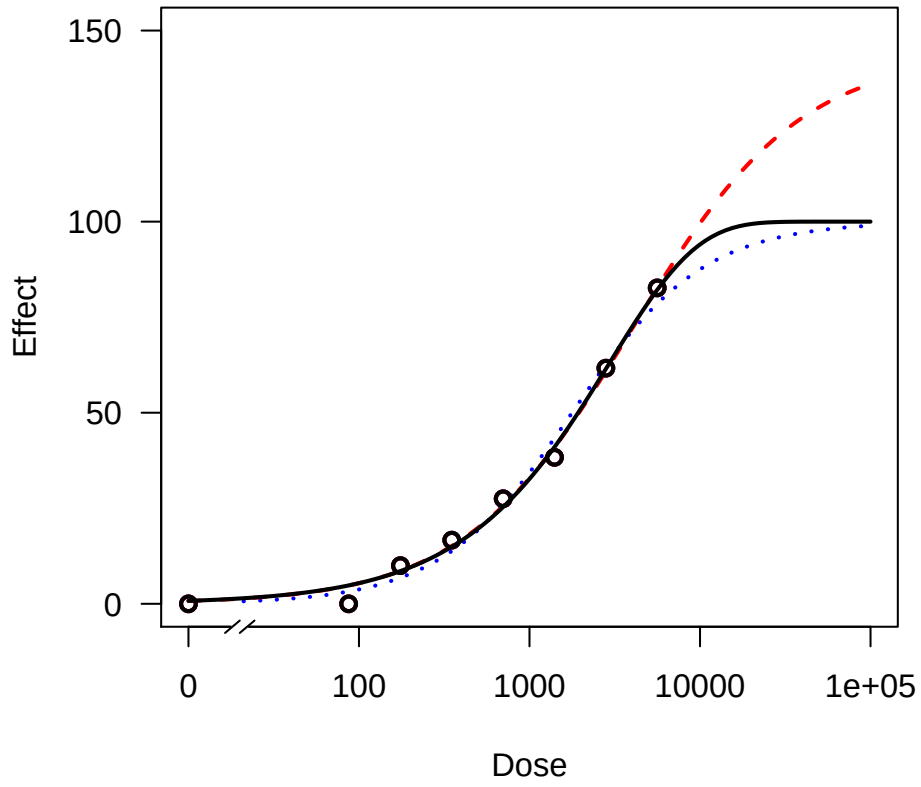

Figure 11: Log-logistic regression fit of *vulpia* without restriction (red line) and with restriction (blue curve) on the upper limit. The Weilbull1 curve `(W2.3)` (black line) also with restriction on the upper limit fitted "best".

7 Hormetic curves

Until now, we have exclusively looked at monotonically decreasing or increasing dose-response curve models. When we experience stimulation of responses at very low concentrations of herbicides or other xenobiotics, however, we need to consider other models for dose-response curves.

An experiment with barley grown in a hydroponic solution with a herbicide showed hormesis at low doses. We specify a dose-response model `fct=CRS.4c()`, which is an extension of the three-parameter log-logistic model (see (**?**) for the details). You can do the familiar `?CRS.4c` search to find out about the dose-response curve.

```
> leaflength.m1 <- drm(DW~Dose, data=leaflength, fct=CRS.4c())
> summary(leaflength.m1)

Model fitted: Cedergreen-Ritz-Streibig (4 parms)

Parameter estimates:

                Estimate Std. Error    t-value p-value
b:(Intercept)   0.480116   0.031867  15.066385  0.0000
d:(Intercept)  10.806220   1.672060   6.462817  0.0000
e:(Intercept)   0.014556   0.018093   0.804502  0.4261
f:(Intercept) 407.462194 209.912114   1.941109  0.0597

Residual standard error:

 4.047613 (38 degrees of freedom)
```

In the `summary` output a new estimated hormesis parameter `f` can be found. On this particular instance we assume that there is only an upper limit and the lower limit is 0. the `e` parameter now has changed its meaning and has nothing to do with the ED50, and the same applies to `b`, which cannot easily be interpreted. The only familiar parameter left is the estimate of the upper limit at untreated control, `d`. Figure 12 displays the observations and the fitted dose-response curve based on the model fit.
Now one of the reasons for fitting response curves is to obtain an EDx value and this can still be done by with the `ED()`

```
> ED(leaflength.m1,50, interval="delta")

Estimated effective doses
(Delta method-based confidence interval(s))

     Estimate Std. Error   Lower Upper
1:50  57.8866    30.7307 -4.3245 120.1
```

```
> plot(leaflength.m1,broken=TRUE,
+ ylab=" Produced leaf length(cm)",
+ xlab="Metsulfuron-methyl(mg/l)")
```

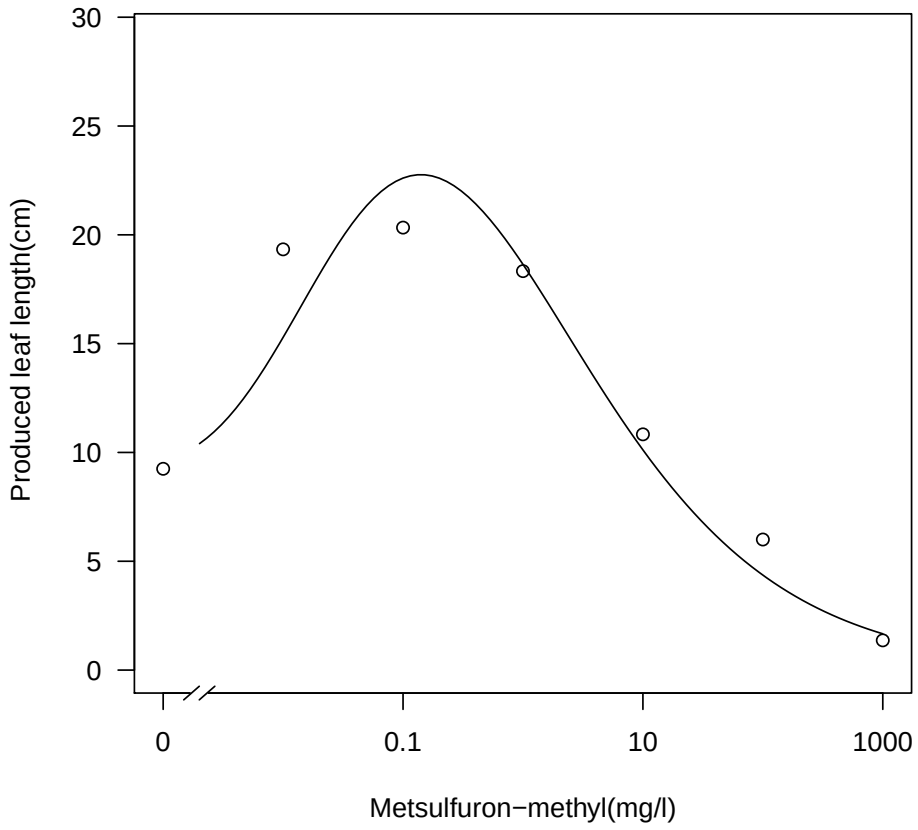

Figure 12: Observations and fitted dose-response curve with a hormesis parameter for the data set `barley`.

To our surprise we find that the ED50 is imprecise and not different from zero. In this case the ED50 is based on the upper limit, `d`.

8 Exponential and Michaelis-Mentens curves

In the studies of plant growth, weed science, enzymes kinetic and ecotoxicology and in many other instances there are other nonlinear curves included in *drc* or other packages, such as the `nlrwr` some of which we will look into. They all are very common in biology.

8.1 Exponential curves

Exponential curves are used to describe the relative growth of plants or the degradation of pesticides in the environment. In `drc` there are two curve with only one or two limits `EXD.2()` and `EXD.3()`. The yield the half life ("T50"), but requires of course that we have a monotonously decreasing curve. The dataframe `decay.TXT`, which is a degradation of a compound in the soil.

```
> Decay<-read.table(file.choose(),header=TRUE)
> Degrade.m1<-drm(y~x,data=Decay,fct=EXD.2())
> summary(Degrade.m1)

Model fitted: Exponential decay with lower limit at 0 (2 parms)

Parameter estimates:

               Estimate Std. Error   t-value p-value
d:(Intercept) 108.13566    5.11370  21.14624       0
e:(Intercept)  12.47100    0.95168  13.10414       0

Residual standard error:

 9.243442 (29 degrees of freedom)
```

The `d` parameter is the intercept with the y-axis and `e` is the T50. This is a very convenient way of calculation the half life of a xenobiotics in the environment, in water and organisms (Fig. 13

If we have initial growth curves without upper limit then an exponential growth curve can be used. The interesting parameter here is the relative rate of change over time, k, We have to launch the add-on package `nlrwr` with the function `SSexp()` where the parameter is the rate constant that can either positive or negative and if negative one can calculate "T50" .

```
> library(nlrwr)
```

and use the function `SSexp()` with the standard nonlinear regression function `nls()`

```
> RGRcurve.m1<-nls(RGR~SSexp(Day,RGR0,k),data=RGRcurve)
> summary(RGRcurve.m1)
```

```
> plot(Degrade.m1, log="")
```

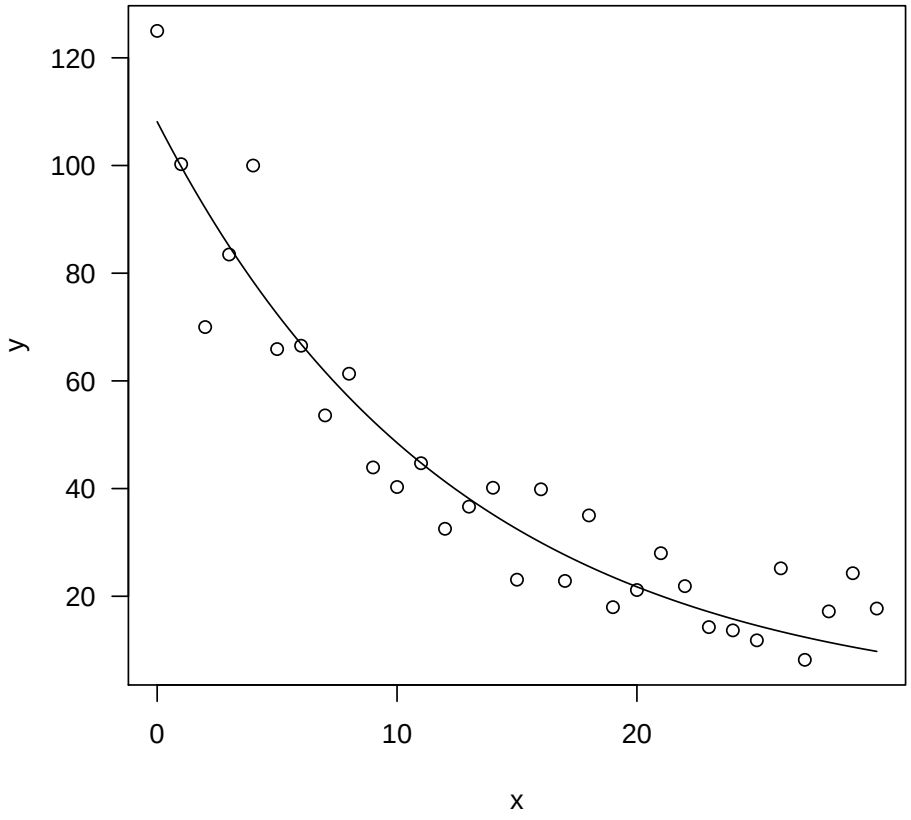

Figure 13: Exponential decay curve, note the argument `log=""` in the plot function. It means that the default logarithmic x-axis is not used.

```
Formula: RGR ~ SSexp(Day, RGR0, k)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
RGR0  0.16372    0.01421   11.52 4.03e-14 ***
k     3.76454    0.17550   21.45  < 2e-16 ***
---
Signif. codes:  0 Ś***Š 0.001 Ś**Š 0.01 Ś*Š 0.05 Ś.Š 0.1 Ś Š 1

Residual standard error: 0.101 on 39 degrees of freedom
```

```
Number of iterations to convergence: 2
Achieved convergence tolerance: 7.896e-06
```

The intercept with y-axis at x=0 is the `RGR0` and and the relative rate of change is `k` However, in this instance we do not have an easy way of plotting the regression line but to get a smooth line we can use the `curve()` function as shown below.

```
> plot(RGR~Day, data=RGRcurve)
> curve(summary(RGRcurve.m1)$parameters[1]*
+ exp(x/summary(RGRcurve.m1)$parameters[2]),0,8,100,add=TRUE)
```

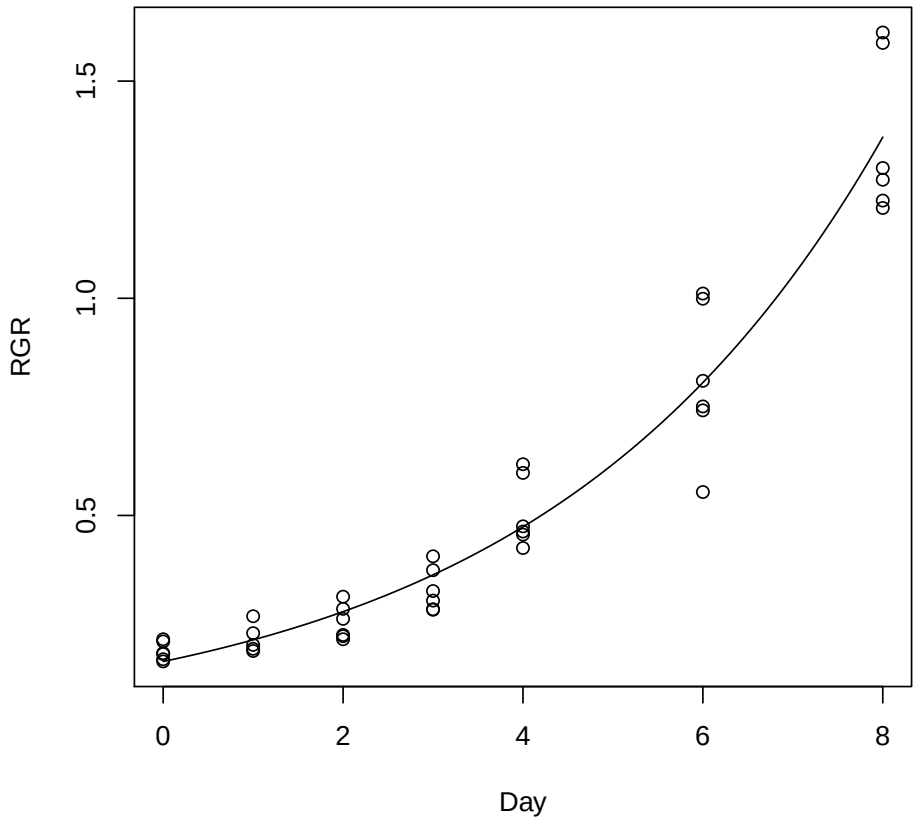

Figure 14: Exponential growth curve

There are numerous other exponential curves but the basic ones are the ones given above. However, it is only the `drm()` function in *drc* that has automated the graphs for

the regression objects. With the `nls()` function we have to do a little extra to get smooth regression curves. In Fig. 14we extract the parameters form the `summary` and write the equation.

8.2 Michaelis-Mentens curves

The general Michaelis-Mentens curve is a very versatile regression curve used in numerous disciplines and among enzymology and Weed Science where it is better known and the rectangular hyperbolic yield loss curve. Whether it is called either names is a question of where you work. If you wish to know more about the curve you can write `?yieldLoss`. First we must read the file *Yieldloss.csv*

```
> yieldloss<-read.csv2(file.choose())
> head(yieldloss)

   Weed Density Yield.loss
1 Weed1       0   0.000000
2 Weed1       4   9.635237
3 Weed1       9  14.246387
4 Weed1      15  14.659326
5 Weed1      22  16.861666
6 Weed2       0   0.000000

> yieldloss.m1 <- drm(Yield.loss~Density, Weed,
+  data = yieldloss, fct = MM.2())
>  summary(yieldloss.m1)

Model fitted: Michaelis-Menten (2 parms)

Parameter estimates:

        Estimate Std. Error  t-value p-value
d:Weed1 19.49695    1.06249 18.35021  0.0000
d:Weed2 28.69350    2.40180 11.94666  0.0000
e:Weed1  3.92425    0.79506  4.93577  0.0026
e:Weed2 11.24707    2.09069  5.37958  0.0017

Residual standard error:

 0.6268349 (6 degrees of freedom)
```

The `summary` shows the parameters, the upper limit `+d:Weed1+`and `d:Weed2` for the two weeds, whilst `+e:Weed1`and e:Weed2+ are the densities that reduce the yield loss by 50% in relation to the upper limits, This is a parallel to the ED50 with the log-logistic curve

```
> plot( yieldloss.m1,log="",xlim=c(0,100),
+ ylim=c(0,30))
```

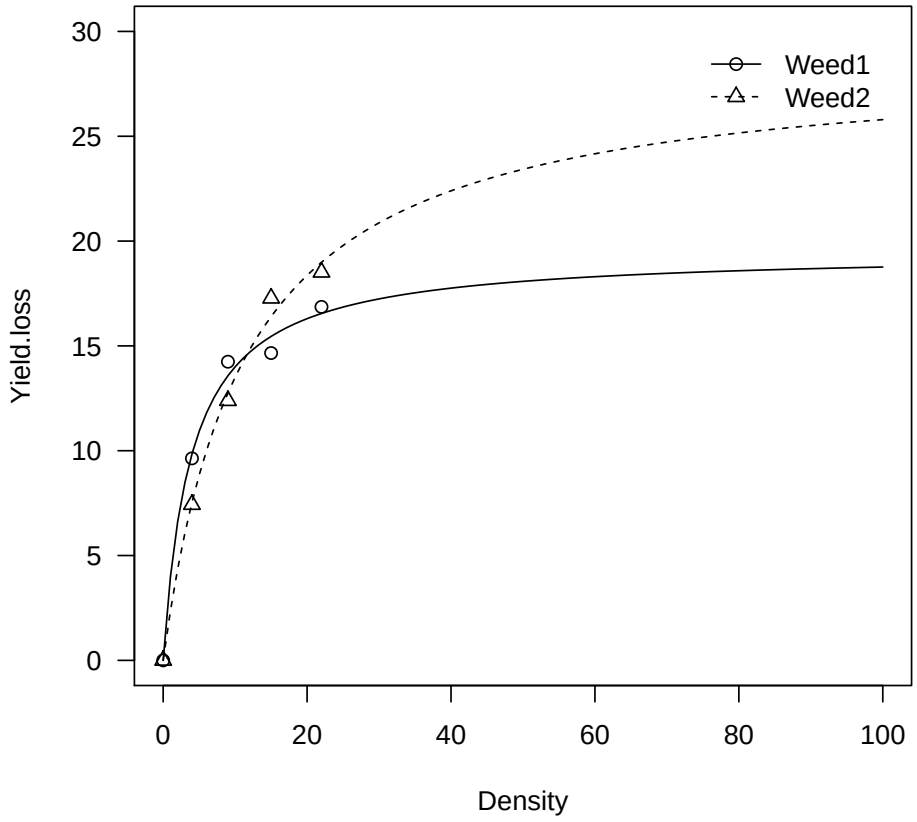

Figure 15: the Michaelis- Mentens curve for the effect of the two weed species on yield of a crop

(Fig. 15. However in Weed Science you are more interested in the competitive ability of a weeds against a crop.

In weed science, however, you usually use the rectangular hyperbolic curve to assess the yieldloss, but it is essentially the same as a common Michaelis-Mentens equation.

```
> yieldLoss(yieldloss.m1 , "as")

Estimated A parameters
(Asymptotically-based confidence interval(s))
```

```
      Estimate Std. Error   Lower  Upper
Weed1  19.4969     1.0625 16.8971 22.097
Weed2  28.6935     2.4018 22.8165 34.571


Estimated I parameters
(Asymptotically-based confidence interval(s))

      Estimate Std. Error   Lower  Upper
Weed1  4.96832    0.76744 3.09046 6.8462
Weed2  2.55120    0.27343 1.88214 3.2203
```

In order to understand the output you could look at the help page `?yieldLoss()` function. Note that the upper limits area exactly the same whether you look at the results using `MM.2()` argument in the `drm()` function of the function `YieldLoss()`. The initial slopes `I` of the curves gives a good indication of which of the two weeds are mist competitive in this particular crop. On the basis of the Michalis-Mentens function we now reparameterize it to the a rectangular model an derive the initial slope is now the one which has been calculated, It is interesting to see that the two initial slopes are different from each other, very little overlap of the confidence intervals.

9 Non linear regression with binomial endpoint

Survival of insect or germination in response to pesticides is a classical experiment. For each dose we have a total number of insects or seeds and the number of insects or seeds affected by the the pesticide. Now we do not work with the normal distribution but the binomial distribution. The data can still be analysed with the function `drm()`, but now we have binomially distributed data the properties of which requires that the response is the proportion of affected divided by the total number of individuals, and two new arguments are included in the `drm()` function, `weights=` and `type="binomial"`.

Survival of insects was examined for various concentrations of the insecticide rotenone. For each concentration we have the number of experimental insects affected after a certain time period (recorded in the variable `affected`) out of the total number of insects receiving the concentration (recorded in the variable `total`). The data is stored in `finney71`

```
> finney71

  dose total affected
1 10.2    50       44
2  7.7    49       42
3  5.1    46       24
4  3.8    48       16
5  2.6    50        6
6  0.0    49        0
```

Fitting the log-logistic model, which in synonym with logistic regression in case of binomially distributed data, yields the parameters below.

```
> model.finney71 <- drm(affected/total~dose, weights=total,
+ data=finney71, fct=LL.2(), type="binomial")
> summary(model.finney71)

Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 and upper limit a

Parameter estimates:

               Estimate Std. Error  t-value p-value
b:(Intercept) -3.10363    0.38773 -8.00472       0
e:(Intercept)  4.82890    0.24958 19.34845       0
```

The argument `fct=LL.2()` specifies a two-parameter log-logistic model model where the lower limit is fixed at 0 and the upper limit is fixed at 1. We now only have to estimate two parameters, the relative slope `b` and the `e`, LD50 (Lethal dose required to affect 50% of the individuals). Note that the argument `type` indicates the type of response to be analysed. It was not necessary to specify this argument in the examples dealing with

continuous endpoints as the default is `type="continuous"`. A plot of the original data and the fitted dose-response curve can give a visual impression of how well the model fits fits to the data, and with such a small dataframe it is only possibly the visually assess the model. The model seems to provide an acceptable fit (Fig. 16.

```
> plot(model.finney71, bp=.5,conName="control",broken=TRUE, legend=FALSE)
```

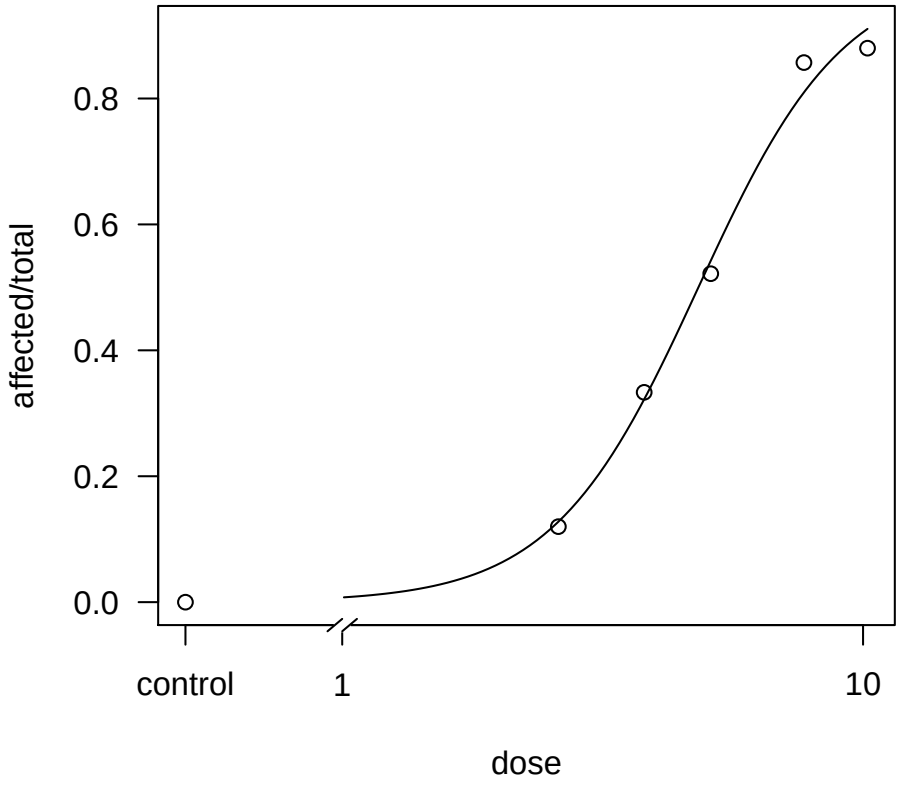

Figure 16: Dose-response curve fitted to data from Finney (1971).

Interestingly, whether you fit continuous or binomial data the estimated parameters are approximately normally distributed, so there are no differences in in the use of the parameters and you can get confidence intervals of the LD50.

```
> ED(model.finney71, 50, interval="delta")

Estimated effective doses
(Delta method-based confidence interval(s))
```

```
     Estimate Std. Error   Lower  Upper
1:50  4.82890    0.24958 4.33974 5.3181
```

9.1 An example with an upper limit

We use the dataframe `earthworms` obtained from a toxicity test, counting numbers of earthworms staying in toxic earth (not migrating to neighbouring clean earth).

A look at the data (try typing `earthworms[1:10,]`) clearly shows that earthworms migrate even though the earth is not toxic (dose is 0). Therefore, it is not appropriate to use a model which assumes a a response range between 0% and 100% but more a and upper limit at 50contaminated at dose 0. We will use a model where the upper limit is a parameter to be estimated. The model is specified as follows.

```
> model1.upper <- drm(number/total~dose, weights=total, data=earthworms,
+ fct=LL.3(), type="binomial")
>
```

The `fct` argument `LL.3()` produces a three-parameter logistic model with the lower limit fixed at 0.

```
> summary(model1.upper)

Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 (3 parms)

Parameter estimates:

              Estimate Std. Error  t-value p-value
b:(Intercept) 1.505679   0.338992 4.441641   0e+00
d:(Intercept) 0.604929   0.085800 7.050498   0e+00
e:(Intercept) 0.292428   0.083895 3.485636   5e-04

>
```

From the `summary` output we see that the estimate of the upper limit is close to the theoretical 0.5. The fitted dose-response curve is displayed in Figure 17. It is by definition symmetric, and it seems that the response values are not following the symmetric curve very well, so an asymmetric model may be more appropriate.

9.2 Multiple dose-response curves

Consider a dataframe where numbers of immobile (versus mobile) daphnids (out of a total of 20 daphnids) has been recorded for several concentrations, both after 24 hours and again after 48 hours. First, we look at the dataframe, the first 6 observations and the last 6 observations in `daphnids`.

```
> plot(model1.upper, broken=TRUE,conName="Control")
>
```

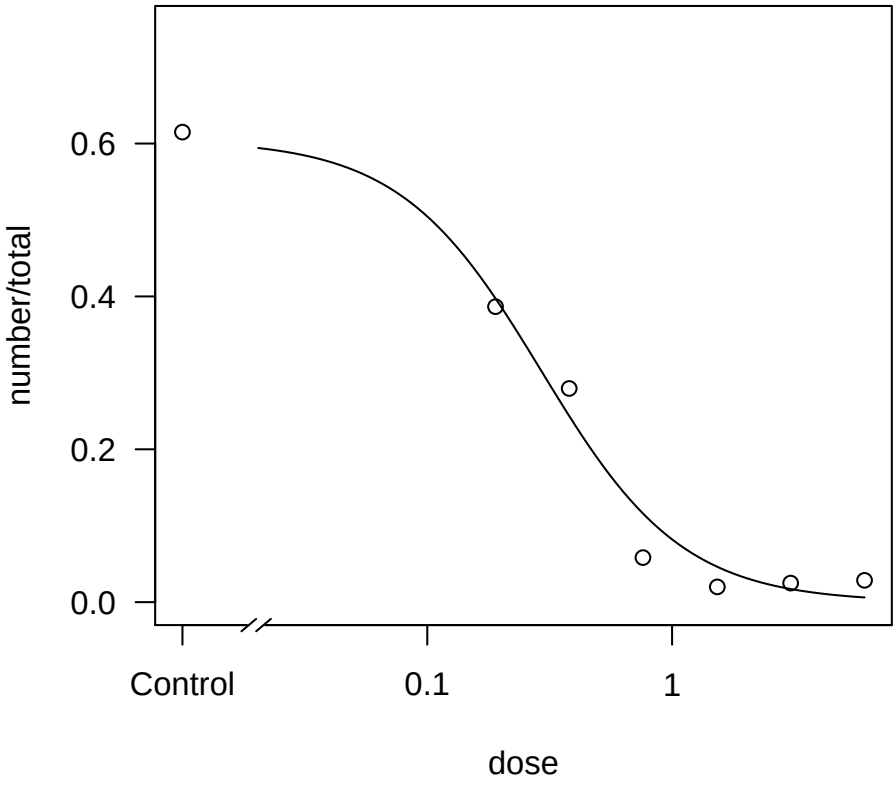

Figure 17: Fitted dose-response curve with upper limit being below 100%.

```
> head(daphnids)

     dose no total time
1  105.00  0    20  24h
2  400.07  1    20  24h
3  600.10  3    20  24h
4 1199.20  3    20  24h
5 1999.33  5    20  24h
6 3198.52  6    20  24h

> tail(daphnids)
```

```
      dose no total time
11  600.10  6    20  48h
12 1199.20  8    20  48h
13 1999.33 11    20  48h
14 3198.52 16    20  48h
15 5596.91 18    20  48h
16 9595.57 20    20  48h
```

The data consist of two curves, one obtained at 24 hours and another obtained at 48 hours after treatment. The initial model assumes that the two curves have different slopes (`b`) and LD50 values (`e`). The model will have a total of four parameters (two `b`s and two `e`s).

```
> model1.daphnids <- drm(no/total~dose, time, weights=total, fct=LL.2(),data=daphnic
> summary(model1.daphnids)

Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 and upper limit a

Parameter estimates:

        Estimate Std. Error    t-value p-value
b:24h   -1.17384    0.22236   -5.27909       0
b:48h   -1.84968    0.27922   -6.62436       0
e:24h 5134.03344 1056.74197    4.85836       0
e:48h 1509.06539  187.76008    8.03720       0

> modelFit(model1.daphnids)

Goodness-of-fit test

            Df Chisq value p value

DRC model   12      13.873  0.3089
```

The second variable, `time`, specifies the variable that divides the data into separate curves and `modelFit` test for lack of fit. The test indicates that the model is appropriate.

It seems that the slopes (the `b` parameters) are rather similar (judged by the estimated standard errors). If similar `b`s the the two curves are identical in all parameters except the LD50s. We can test this formally by using the function `compParm` which can calculate pairwise differences (or ratios) for a chosen model parameter, here the slope parameter `b`).

```
> compParm(model1.daphnids, "b", "-")

Comparison of parameter 'b'

        Estimate Std. Error t-value p-value
24h-48h  0.67584    0.35694 1.89341  0.0583
```

The argument "-" indicates the differences between parameter estimates are to be calculated. The test produced by compParm yields an approximate t test, which in this particular instance is not significantly different from 0. We can also use a likelihood ratio test to see whether or not the slopes are equal. The likelihood ratio test is a measure of the distance between two models. If the distance is small according to an appropriate reference (a χ^2 distribution), the conclusion is that the two models provide similar descriptions of the data, favouring the smaller model, that is the model with fewer parameters, as it provides the simplest description of the data.

The model with a common bs si shown below.

```
> model2.daphnids <- drm(no/total~dose, time, fct=LL.2(),
+ weights=total, data=daphnids,
+ type="binomial", pmodels=data.frame(1,time))
```

The likelihood ratio test is calculated by anova.

```
> anova(model2.daphnids, model1.daphnids)
```

```
1st model
 fct:     LL.2()
 pmodels: 1, time
2nd model
 fct:     LL.2()
 pmodels: time (for all parameters)

ANOVA-like table

          ModelDf  Loglik Df LR value p value
1st model       3 -27.854
2nd model       4 -26.000  1   3.7073  0.0542
```

The likelihood ratio test supports the conclusion, based on the t test we can assume common bs. It means that the relative potency between the two curves are the same at any response levels.

```
> summary(model2.daphnids)
```

```
Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 and upper limit a

Parameter estimates:

               Estimate Std. Error    t-value p-value
b:(Intercept)   -1.49854    0.17339   -8.64253       0
e:24h         4615.83991  708.69555    6.51315       0
e:48h         1492.40061  211.28616    7.06341       0
```

```
> SI(model2.daphnids,c(50,50))

Estimated ratios of effect doses

              Estimate Std. Error t-value p-value
24h/48h:50/50  3.09290    0.64967 3.22148  0.0013


> plot(model1.daphnids, lty=c(1,1), ylim=c(0, 1.2) )
> plot(model2.daphnids, add = TRUE, type = "none", legend = FALSE, col=c(2,2), lty=c
```

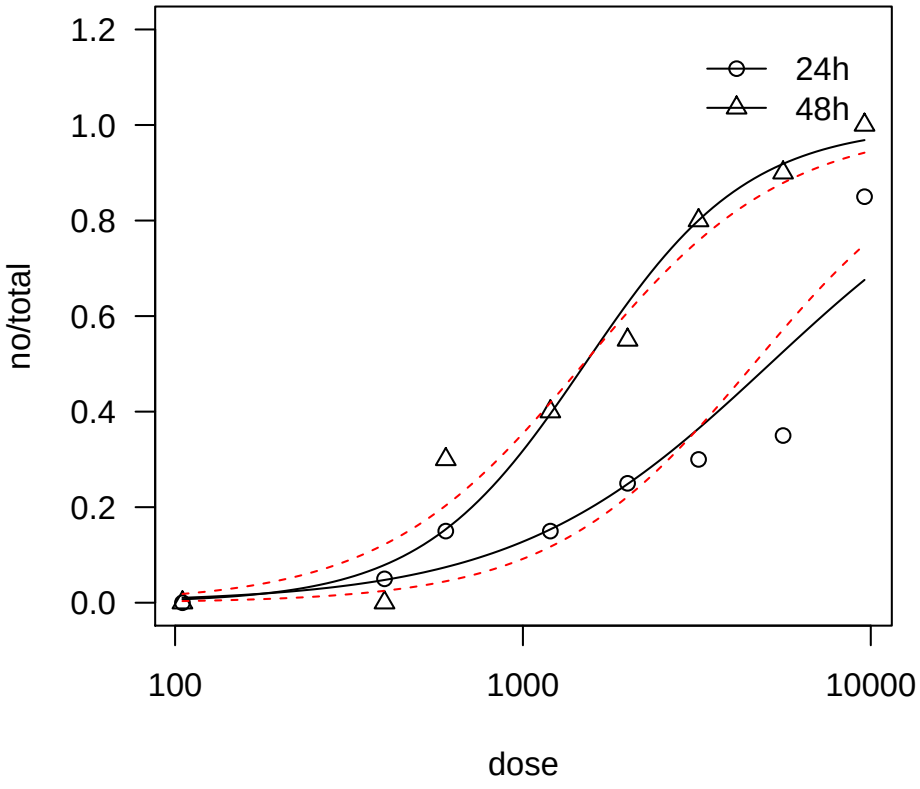

Figure 18: Different `bs` and common `b` (red dashed line).

A visual comparison of the two models is displayed in Figure 18. The binomial models are also useful to describe germination in course of times or concentration of xenobiotics. In weed science literature and elsewhere there is a relaxed relationship to use binomial

regressions, in fact it is more an exception to the rule of converting binomial data to percentage and run an ordinary regression analysis . The majority uses the percentage of germinated seeds or surviving plants. By doing so you get almost the same parameters, but the standard errors are wrong. It means that if you will take advantage of being able to study differences between the vigor of various seed lots you do it with the wrong standard error. If you neglect the structure of the data in toxicology and ecotoxicology your safety factors and relative potencies rest on erroneous uncertainties. And please remember it does not matter how many replications you use, the standard errors are still be wrong if you do not use the proper binomial distribution.

10 Statistical considerations

In the past sections we have not lingered much over the statistics and the statistics assumption behind the fitting of regressions. We only looked at the parameters and and their standard errors and found out that in certain instances the lower limit of a log-logistic curve was not significantly different from zero. Also we introduced a test for lack of fit where we test if a regressions were just as good as an ordinary ANOVA. The most general model in most instances is the ANOVA and threfore it is convenient to uses this model as reference. However, there are some prerequisites that are important when doing any statistical analysis. This applies to continuous variables The three most important ones are:

- The mean function is correct
- Variance is homogeneous
- Residuals must be normally distributed around 0.0.

The obvious prerequisite is the use of a correct model, or as we can say the "best" model that do not violate the relationship between the response and independent variable. The choice of regression model is on the discretion of the biologist.

The precision of parameters and their associate standard errors is only valid if the variance is homogenous and the residuals are normally distributed. The reason for this is that we want the statistical parameters to be normally distributed so we can use them in test of various hypotheses of biological importance. Consequently, most part of this section has nothing to do with biology behind the regression, but only to explore if the statistical prerequisites are partly fulfilled.

10.1 Heterogeneous variance

The problem with the heterogeneous variance in Figure 19 is a statistical problem not a biological one. The parameters of the regression model is used to test or biologically assess various hypotheses, but in order to do so the underlying assumption for doing a regressions analysis in the first place must be full filled. If the homogeneity of variance is not met then the parameter estimates and particularly their standard errors may not be correct and any further statistical test could be futile. In order to unravel the problem we begin with a simple analysis of residuals in Figure 19 very common in this kind of experiments. The original fit is in Fig. 6 where the upper an lower limits are the same for the wto species.

The distribution of the residuals in Fig. 19 has a funnel type shape with the most narrow distribution around the line of zero at low `DryMatter`, whilst at the higher predicted values there are a wider distribution around the zero line. This problem of heterogeneity of variance can be dealt with by using a transform both sides technique. *drc* has a build in function `boxcox()`that builds upon already fitted curves. Notice if you wish to know more about the `boxcox()` function in *drc*, you can write `?boxcox.drc`.

```
> S.alba.m2BC <- boxcox(S.alba.m2,method="anova")
> summary(S.alba.m2BC)

Model fitted: Log-logistic (ED50 as parameter) (4 parms)

Parameter estimates:

               Estimate Std. Error   t-value p-value
b:Bentazone    4.838636   0.927240  5.218322       0
b:Glyphosate   1.944311   0.236471  8.222178       0
c:(Intercept)  0.682591   0.028768 23.727039       0
d:(Intercept)  3.862611   0.106186 36.375984       0
e:Bentazone   28.396147   1.874598 15.147857       0
e:Glyphosate  65.573335   5.618945 11.670043       0

Residual standard error:

 0.1558947 (62 degrees of freedom)

Non-normality/heterogeneity adjustment through optimal Box-Cox transformation

Estimated lambda: 0.101
Confidence interval for lambda: [-0.126, 0.331]
```

The determination of find the best lambda is see in Fig. 20.

By comparing the summary for the fits above with the original one on page 17 the differences are not dramatic between the two regressions, and the question is whether use of correction with Boxcox does matter at all. When it comes to the normality of the residuals we can do a quantile plot (Fig. 21)

```
qqnorm(resid(S.alba.m2BC))
qqline(resid(S.alba.m2BC))
```

From the distribution there are no reason to believe that the the normality is heavily violated

The transform both side technique is not changing the relationship between response and x-variable. If only the right hand side was transformed the relationship would be violated. There are other means of taking care of the heterogeneity of variance such of weighting but we will not go into this. It has been dealt with elsewhere Ritz og streibig

```
> plot(resid(S.alba.m2)~predict(S.alba.m2))
> abline(h=0)
```

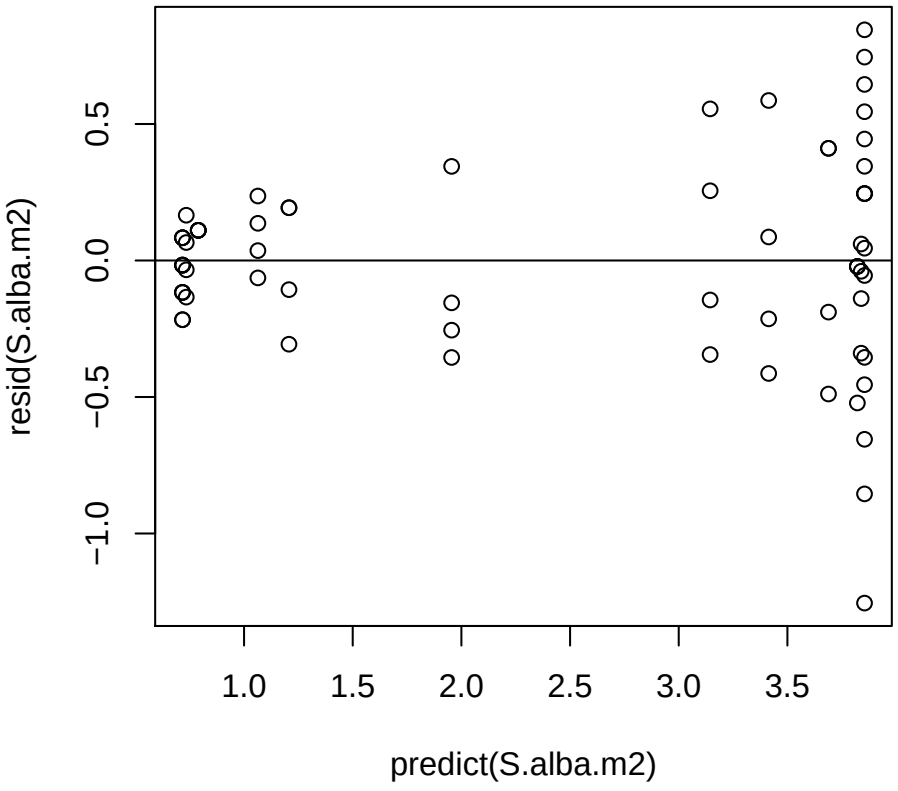

Figure 19: Distribution of residuals from the regression analysis of the `S.alba` data.

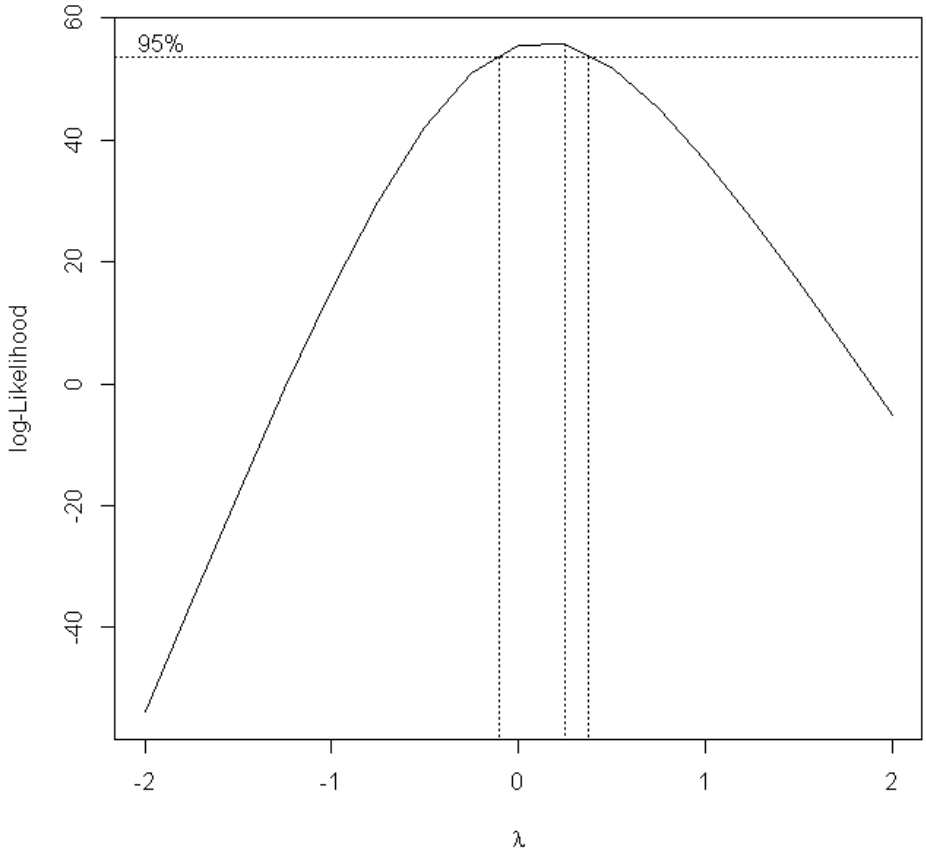

Figure 20: The search for the optimal lambda estimate to ensure homogenerity of variance for the dose-response for the `S.alba` data.

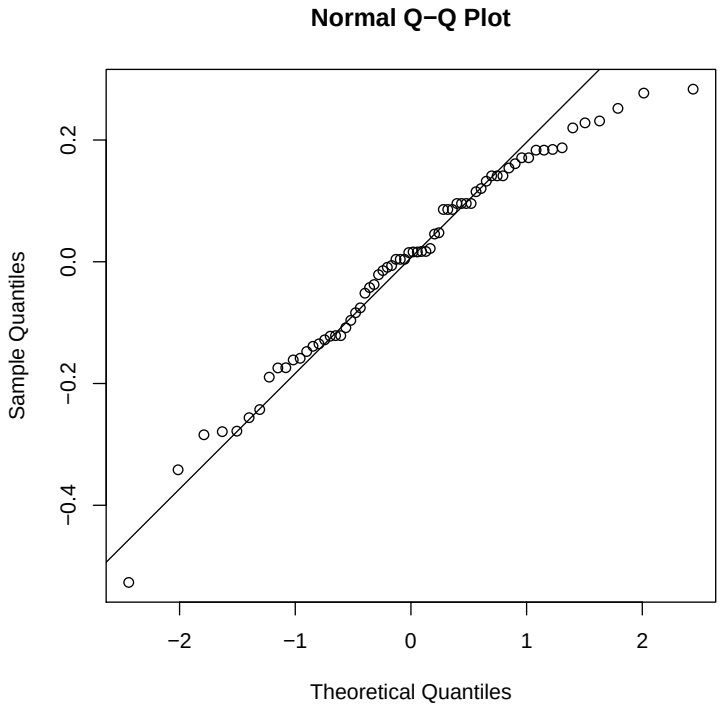

Figure 21: Quantile-quantile plot to illustrate if the residual of responses are normally distributed.