R Notebook: DRC package and dose response modeling

Andrew East

2/23/18

After installing R and Rstudio, install the `drc` package by running the below code in your R console. Alternatively, enter the below code into a script and run in the Rstudio script editor. Creating a script for consistency across projects and time is good practice.

```
install.packages("drc", repos="https://cloud.r-project.org")
```

After the `drc` package is installed, we'll make sure that R knows were going to use it.

```
library(drc)
```

By "library-ing" the package, R will now have available the functions that are included in the package. The key here is that these functions are beyond the `base` package R is default installed with. Additionally, a working knowledge of R and Rstudio download, command line interface, and data import are assumed. This training session will however, included all data needed.

The intent of this notebook is to familiarize a toxicolgist with dose-response modeling in R. It is assumed that they know the details of different dose reponse relationships, their functions, and endpoints.

Initial Data

As with any statistical adventure, we need to start with data. The code below concatenates (`c()`) some data into vectors that will be our pretend toxicity test results. Notice that the data is divided into `dose` and `resp` vectors.

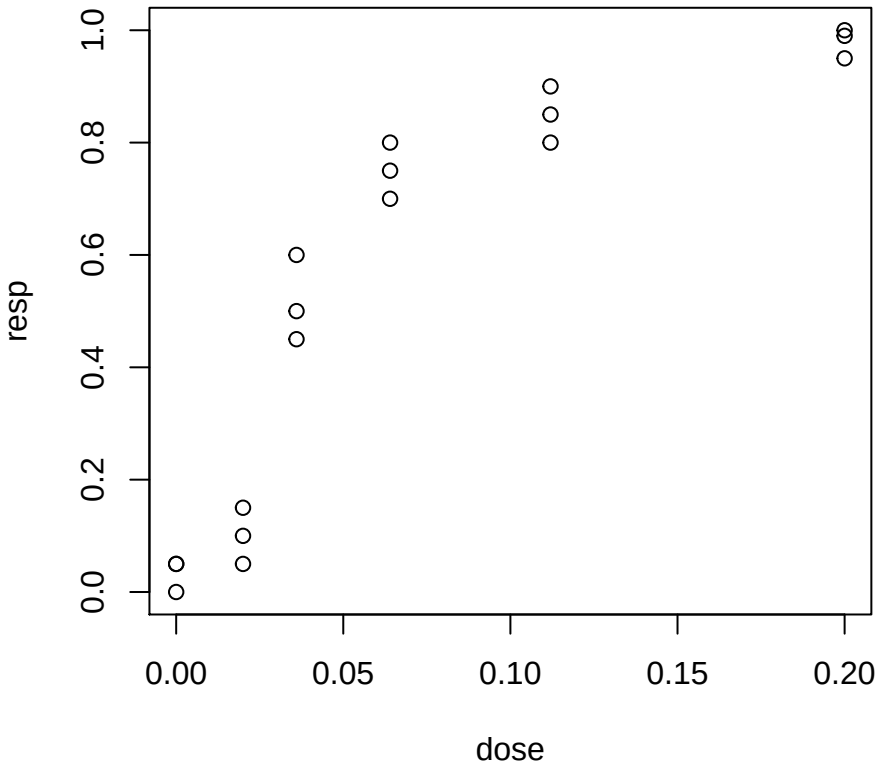
```
dose <- c(0,0,0,
          0.02,0.02,0.02,
          0.036,0.036,0.036,
          0.064,0.064,0.064,
          0.112,0.112,0.112,
          0.2,0.2,0.2)
resp <- c(0,0.05,0.05,
          0.1,0.15,0.05,
          0.45,0.6,0.5,
          0.7,0.75,0.8,
          0.8,0.9,0.85
          ,0.99,1,0.95)
```

Of note, due to this simplified data structure, the treatments (`dose`) are in order and the replicates are grouped.

Data Visualization

To get a simplified sense of the data let's do some plotting. We'll start with a scatterplot.

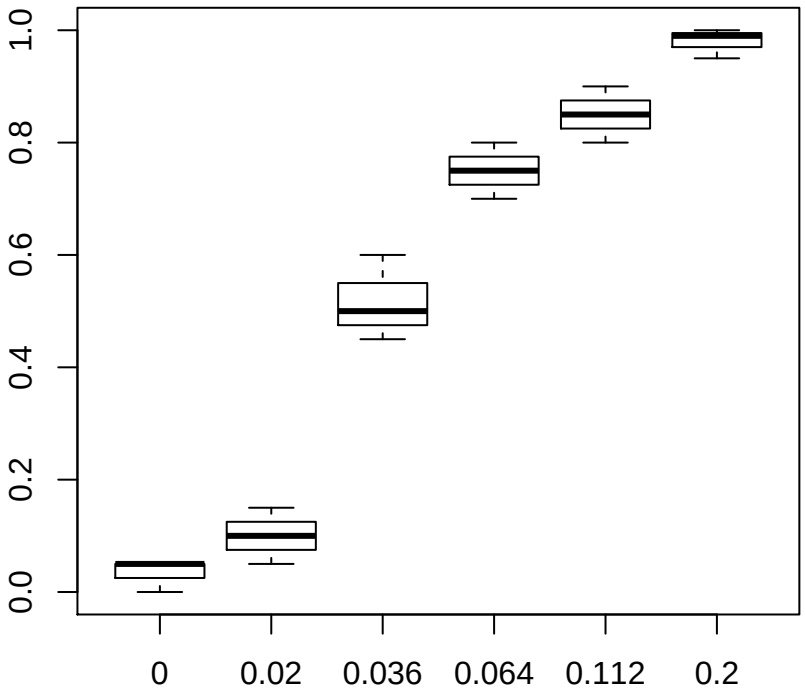
```
plot(resp~dose)
```



Of note is the syntax of the `plot()` function; `Y~X` is the `formula` method. `plot(x,y)` is also possible, but can create issues in more complex function syntax.

Another way to visualize the data is a boxplot. The boxplot method is more useful when relating to policy based toxicity tests. i.e. NOEC and LOEC are determined by an ANOVA type test treating the doses as categorical factors.

```
doseF <- factor(dose)
boxplot(resp~doseF)
```



ANOVA-based Analysis

Accordingly, to continue the ANOVA frame of reference, the R code and output for a simple ANOVA and Tukey HSD test is:

```
aov1 <- aov(resp~doseF)
summary(aov1)
TukeyHSD(aov1)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## doseF        5 2.3539  0.4708     190 5.44e-11 ***
## Residuals   12 0.0297  0.0025
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = resp ~ doseF)
##
## $doseF
##                   diff          lwr       upr     p adj
## 0.02-0      0.06666667 -0.069849888 0.2031832 0.5902767
## 0.036-0     0.48333333  0.346816778 0.6198499 0.0000006
## 0.064-0     0.71666667  0.580150112 0.8531832 0.0000000
## 0.112-0     0.81666667  0.680150112 0.9531832 0.0000000
## 0.2-0       0.94666667  0.810150112 1.0831832 0.0000000
## 0.036-0.02  0.41666667  0.280150112 0.5531832 0.0000032
## 0.064-0.02  0.65000000  0.513483445 0.7865166 0.0000000
## 0.112-0.02  0.75000000  0.613483445 0.8865166 0.0000000
## 0.2-0.02    0.88000000  0.743483445 1.0165166 0.0000000
## 0.064-0.036 0.23333333  0.096816778 0.3698499 0.0010087
```

```
## 0.112-0.036 0.33333333  0.196816778 0.4698499 0.0000336
## 0.2-0.036   0.46333333  0.326816778 0.5998499 0.0000010
## 0.112-0.064 0.10000000 -0.036516555 0.2365166 0.2104895
## 0.2-0.064   0.23000000  0.093483445 0.3665166 0.0011440
## 0.2-0.112   0.13000000 -0.006516555 0.2665166 0.0653217
```

This output tells us that dose 0.02 is not statistically significantly different than the control at an alpha of 0.05.

While this hypothesis testing approach is useful in its simplicity, there will be scenarios where prediction of adverse outcomes will be the desired endpoint. i.e. given a water concentration, what percent of effect would we anticipate. Clearly, some function is needed to interpolate between ttested concentrations.

Quantitative and Predictive Analysis

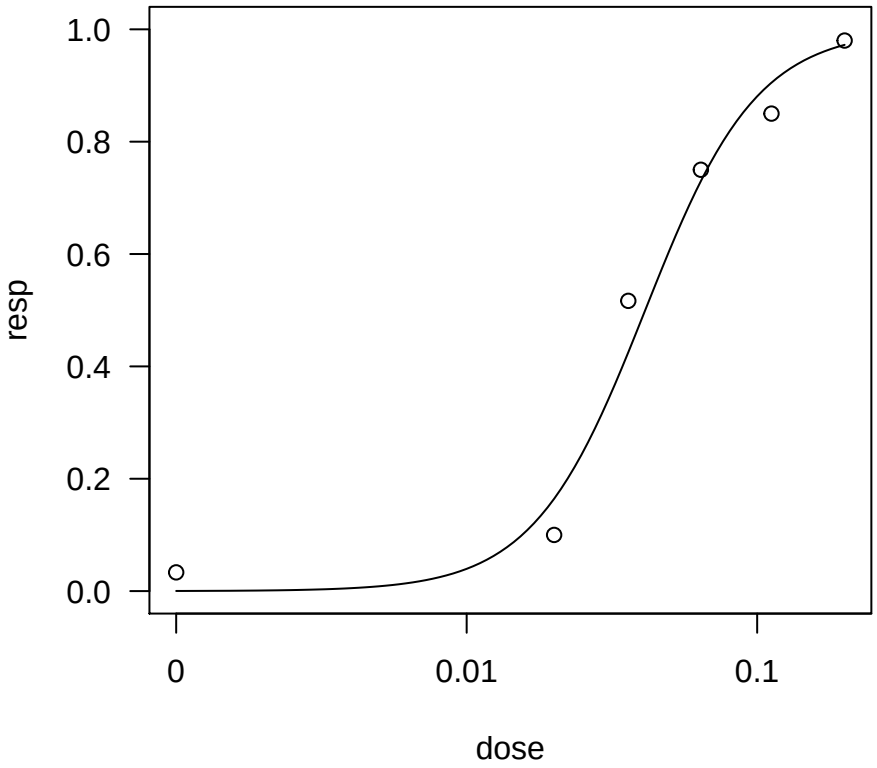
We will start by using the `drm()` function to build a ***d***ose ***r***esponse ***m***odel object that will be used in several other analysis and visualization functions. Always start with the help files `?drm`.

```
?drm
drm1 <- drm(resp~dose, fct=LL.2(), type="binomial")
```

Here, we can see the same `formula` layout as the `plot()` function above, the `fct=` argument is refering to specific mathematical function to use to fit to the data. Here we are using `LL.2()` meaning a two parameter log-logistic function. The type indicates whether data are quantal/binomial (eg. live vs dead) or continuous (eg. mass).

The easiest place to begin is by visualizing the model.

```
plot(drm1)
```

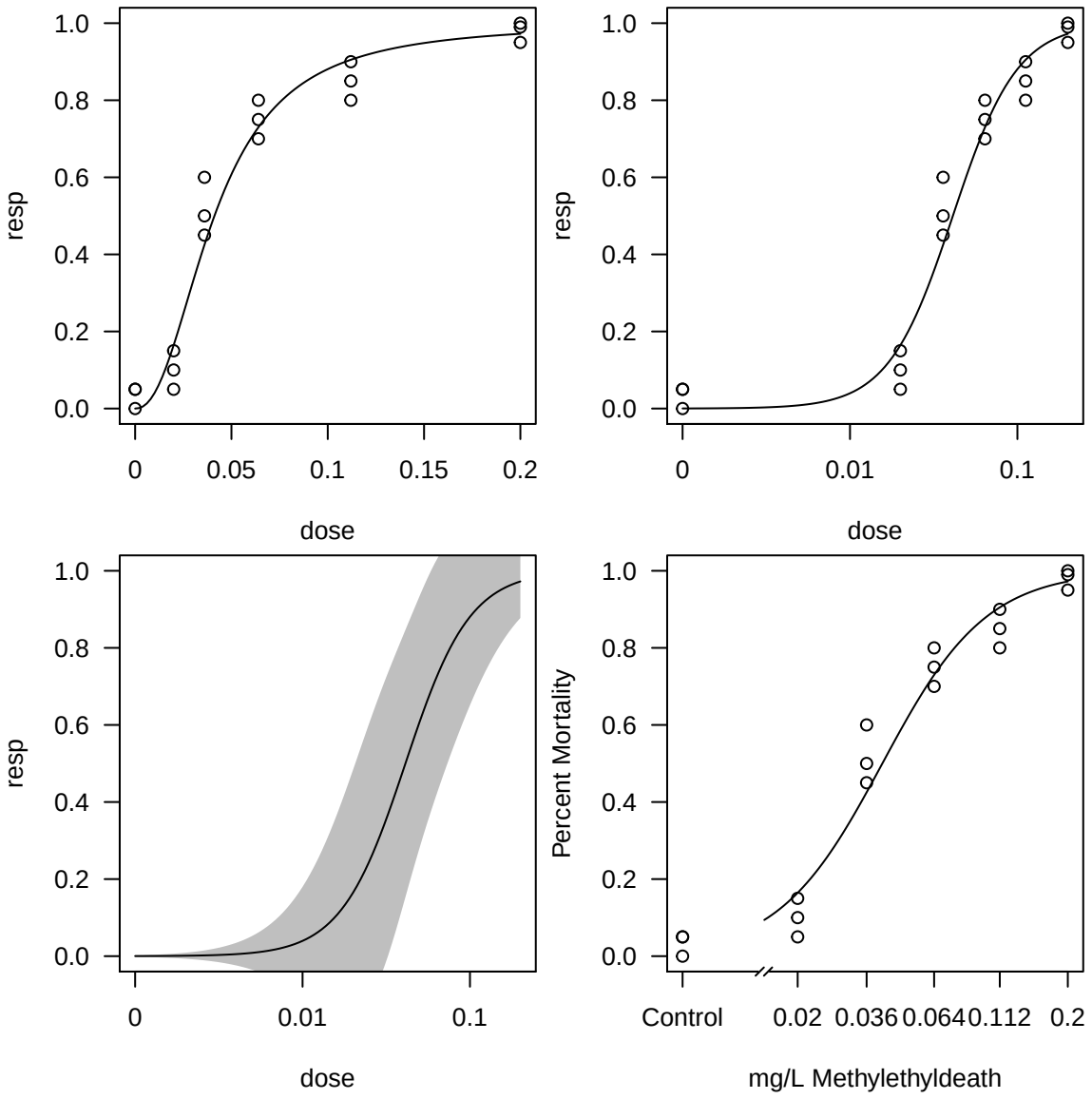


We notice several things about this plot. Firstly, the data have been summarized and only the means are plotted, the x-axis is log transformed, and the model appears to fit the data well by eye. The below plots show

a variety of adjustments to the plotting of a drm object. `type=""` can display `all` the points or `confidence` intervals for instance. `log=""` plots one (or both) of the axes on a log scale. The last plot shows how to label the axes (`xlab="", ylab=""`) and how to adjust the x-axis to bring your control and first dose value a bit closer together by breaking the axis.

Of note, the `par()` function is used to arrange these plots into a grid format (`mfrow=c(#,#)`) and reduces the margins around the plot (`mai=c(#,#,#,#)`) to bring them a bit closer. The graphical parameters are brought back to default settings after the plots so the next plots are normal.

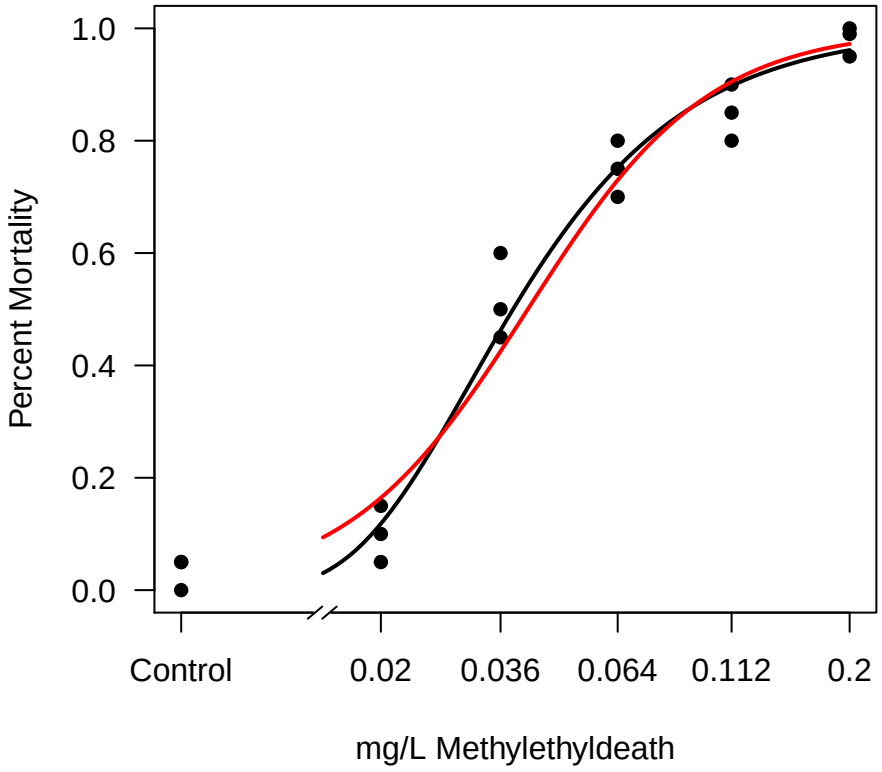
```
par(mfrow=c(2,2), mai=c(0.67,0.67,0.05,0.05))
plot(drm1, type="all", log="")
plot(drm1, type="all", log="x")
plot(drm1, type="confidence")
plot(drm1, type="all", xlab="mg/L Methylethyldeath",
     ylab="Percent Mortality", broken=T,
     bp=0.0075, xt=c(0,0.02,0.036,0.064,0.112,0.2),
     xtlab=c("Control",0.02,0.036,0.064,0.112,0.2))
```



```
par(mfrow=c(1,1), mai=c(1,1,1,1))
```

The next thing we'll want to do is establish whether the two parameter log-logistic function is the "best" model to explain these data. the `mselect()` function is used to compare the fits of the available models and rank them according to Aikaike's Information Criteria (AIC). To find out which models are available, the code `getMeanFunctions()` will display all that are available in the `drc` package. Once the best performing model is determined, create a new drm object and compare it visually against the prior LL.2() model.

```
getMeanFunctions()
mselect(drm1, list(LL.3(), W1.2()))
drmw12 <- drm(resp~dose, fct=W1.2(), type="binomial")
plot(drmw12, type="all", pch=16, lwd=2,
     xlab="mg/L Methylethyldeath", ylab="Percent Mortality",
     broken=T, bp=0.0075, xt=c(0,0.02,0.036,0.064,0.112,0.2),
     xtlab=c("Control",0.02,0.036,0.064,0.112,0.2))
plot(drm1, type="none", add=T, col="red", lwd=2, broken=T, bp=0.0075)
```




```
## Log-logistic (ED50 as parameter) with lower limit at 0 and upper limit at 1
## (2 parameters)
## In 'drc':  LL.2
##
## Log-logistic (ED50 as parameter) with lower limit at 0
## (3 parameters)
## In 'drc':  LL.3
##
## Log-logistic (ED50 as parameter) with upper limit at 1
## (3 parameters)
## In 'drc':  LL.3u
##
## Log-logistic (ED50 as parameter)
```

```
## (4 parameters)
## In 'drc':  LL.4
##
## Generalized log-logistic (ED50 as parameter)
## (5 parameters)
## In 'drc':  LL.5
##
## Weibull (type 1) with lower limit at 0 and upper limit at 1
## (2 parameters)
## In 'drc':  W1.2
##
## Weibull (type 1) with lower limit at 0
## (3 parameters)
## In 'drc':  W1.3
##
## Weibull (type 1)
## (4 parameters)
## In 'drc':  W1.4
##
## Weibull (type 2) with lower limit at 0 and upper limit at 1
## (2 parameters)
## In 'drc':  W2.2
##
## Weibull (type 2) with lower limit at 0
## (3 parameters)
## In 'drc':  W2.3
##
## Weibull (type 2)
## (4 parameters)
## In 'drc':  W2.4
##
## Brain-Cousens (hormesis) with lower limit fixed at 0
## (4 parameters)
## In 'drc':  BC.4
##
## Brain-Cousens (hormesis)
## (5 parameters)
## In 'drc':  BC.5
##
## Log-logistic (log(ED50) as parameter) with lower limit at 0 and upper limit at 1
## (2 parameters)
## In 'drc':  LL2.2
##
## Log-logistic (log(ED50) as parameter) with lower limit at 0
## (3 parameters)
## In 'drc':  LL2.3
##
## Log-logistic (log(ED50) as parameter) with upper limit at 1
## (3 parameters)
## In 'drc':  LL2.3u
##
## Log-logistic (log(ED50) as parameter)
## (4 parameters)
## In 'drc':  LL2.4
```

```
##
## Generalised log-logistic (log(ED50) as parameter)
## (5 parameters)
## In 'drc':  LL2.5
##
## Asymptotic regression with lower limit at 0
## (2 parameters)
## In 'drc':  AR.2
##
## Shifted asymptotic regression
## (3 parameters)
## In 'drc':  AR.3
##
## Michaelis-Menten
## (2 parameters)
## In 'drc':  MM.2
##
## Shifted Michaelis-Menten
## (3 parameters)
## In 'drc':  MM.3
##
##         logLik       IC Lack of fit
## W1.2 -9.138190 22.27638   1.0000000
## LL.2 -9.219974 22.43995   0.9999999
## LL.3 -9.215527 24.43105   0.9999997
```

Now that we have a model that fits well, let's use this model to make some predictions about toxicity endpoints. For instance, an LC50.

```
lc50 <- ED(drmw12, 50, interval="delta")
```

```
##
## Estimated effective doses
##
##        Estimate Std. Error    Lower    Upper
## e:1:50 0.038286   0.010487 0.017732 0.058839
```

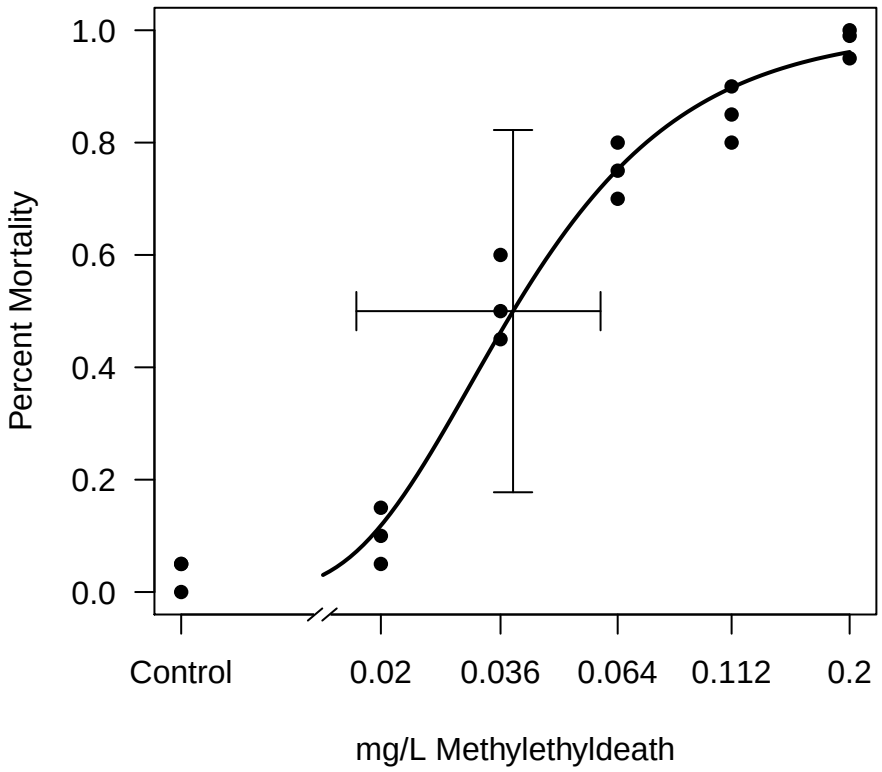
`lc50` gives us an estimate of the dose expected to cause 50% mortality, the standard error of that estimate, and upper and lower confidence intervals. Once we know the concentration, we can also predict the distribution of mortalities expected from exposure to that concentration.

```
mort038 <- predict(drmw12, newdata=data.frame(lc50[1]), interval="confidence")
mort038
```

```
## Prediction      Lower      Upper
##   0.500000   0.177612   0.822388
```

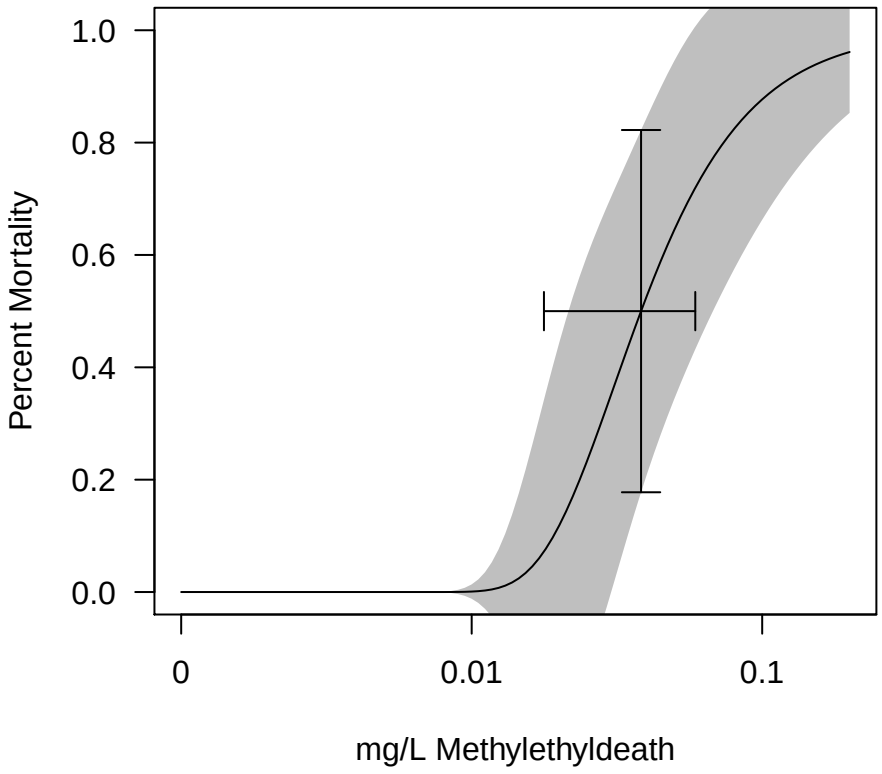
With these values, we can plot a useful visulization of these confidence intervals.

```
plot(drmw12, type="all", pch=16, lwd=2,
     xlab="mg/L Methylethyldeath", ylab="Percent Mortality",
     broken=T, bp=0.0075, xt=c(0,0.02,0.036,0.064,0.112,0.2),
     xtlab=c("Control",0.02,0.036,0.064,0.112,0.2))
arrows(lc50[3],0.5,lc50[4],0.5, length=0.1, angle=90, code=3)
arrows(lc50[1],mort038[2],lc50[1],mort038[3], length=0.1, angle=90, code=3)
```



This plot is essentially what `type="confidence"` is returning and visually describes the interation between the stochastic death and individual threshold mortality models. (see GUTS model for further info)

```
plot(drmw12, type="confidence", xlab="mg/L Methylethyldeath", ylab="Percent Mortality")
arrows(lc50[3],0.5,lc50[4],0.5, length=0.1, angle=90, code=3)
arrows(lc50[1],mort038[2],lc50[1],mort038[3], length=0.1, angle=90, code=3)
```



Compare multiple treatment types

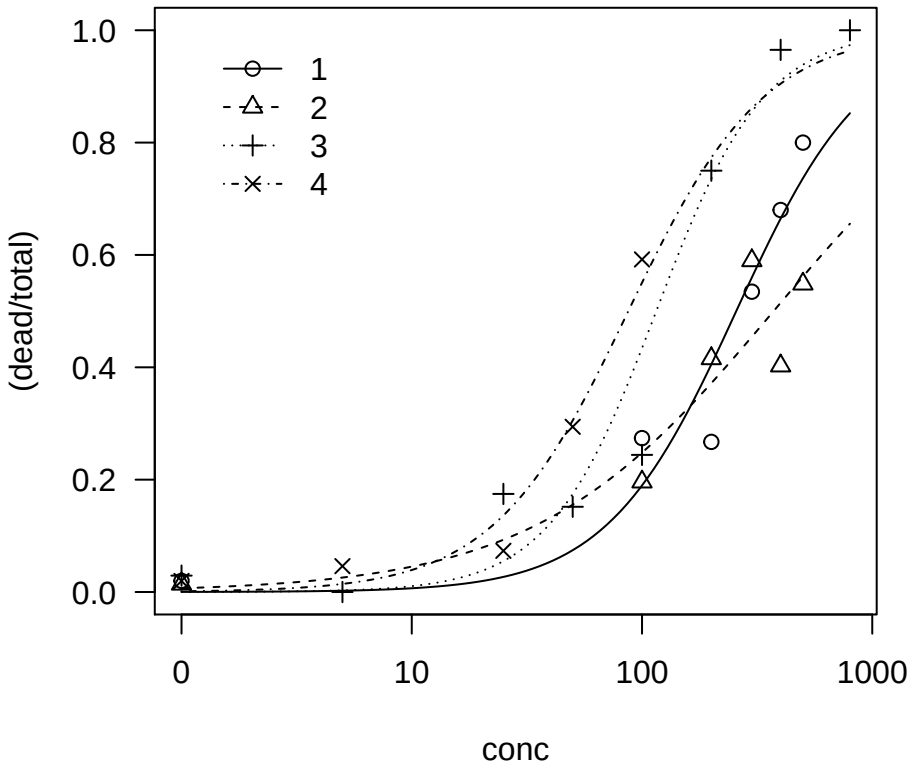
In this section, we're going to use data that comes with the `drc` package and represents the mortality response of multiple different types of selenium.

```
head(selenium)
```

```
##   type conc total dead
## 1    1    0   151    3
## 2    1  100   146   40
## 3    1  200   116   31
## 4    1  300   159   85
## 5    1  400   150  102
## 6    1  500   140  112
```

In this dataframe, there is a concentration vector, a total vector, a dead vector, and a type vector. Concentration represents the dose or exposure concentration, total represents the total number of animals at the beginning of exposure, dead represents the number that died during the exposure, and type represents the type of selenium the animals were exposed to. In this case, type is a group or category.

```
seldrm <- drm((dead/total)~conc, curveid=type, data=selenium, fct=LL.2(), type="binomial")
plot(seldrm, ylim=c(0,1), legendPos=c(5,1))
```




```
summary(seldrm)
```

```
##
## Model fitted: Log-logistic (ED50 as parameter) with lower limit at 0 and upper limit at 1 (2 parms)
##
## Parameter estimates:
##
##      Estimate Std. Error t-value p-value
## b:1  -1.54137    1.87626 -0.8215  0.4114
## b:2  -0.84279    1.72053 -0.4898  0.6242
## b:3  -1.86014    1.36362 -1.3641  0.1725
## b:4  -1.47849    2.01650 -0.7332  0.4634
## e:1 256.59456  162.87417  1.5754  0.1152
## e:2 372.81464  472.51309  0.7890  0.4301
## e:3 115.33042   71.77087  1.6069  0.1081
## e:4  87.17648   93.95583  0.9278  0.3535
```

The above dose response model fits the same function to 4 different response groups defined by `curveid=type`. Accordingly, when we look at LC50 values (below) we can identify that the responses may be different.

```
ED(seldrm, 50, interval="delta")
```

```
##
## Estimated effective doses
##
##        Estimate Std. Error    Lower    Upper
## e:1:50  256.595    162.874  -62.633  575.822
## e:2:50  372.815    472.513 -553.294 1298.923
## e:3:50  115.330     71.771  -25.338  255.999
## e:4:50   87.176     93.956  -96.974  271.327
```

First thing to note is the decrease in resistance as the type increases. Second thing to note is the variation in error. With that in mind it may be worth checking for statistical significance of the difference in estimates doue to the variation.

```
compParm(seldrm, "e", operator="-")
#comparison of inflection point parameter "e" (analogous to lc50)
compParm(seldrm, "b", operator="-")
#comparison of slope parameter "b" (analogous to linear slope around lc50 point)
```

```
##
## Comparison of parameter 'e'
##
##     Estimate Std. Error t-value p-value
## 1-2 -116.220    499.797 -0.2325  0.8161
## 1-3  141.264    177.986  0.7937  0.4274
## 1-4  169.418    188.031  0.9010  0.3676
## 2-3  257.484    477.933  0.5387  0.5901
## 2-4  285.638    481.764  0.5929  0.5532
## 3-4   28.154    118.232  0.2381  0.8118
##
## Comparison of parameter 'b'
##
##      Estimate Std. Error t-value p-value
## 1-2 -0.698582   2.545693 -0.2744  0.7838
## 1-3  0.318776   2.319439  0.1374  0.8907
## 1-4 -0.062874   2.754380 -0.0228  0.9818
## 2-3  1.017358   2.195374  0.4634  0.6431
## 2-4  0.635708   2.650750  0.2398  0.8105
## 3-4 -0.381650   2.434281 -0.1568  0.8754
```

As the above statistical tests show, the difference between model parameters are not significant for any types of selenium treatments. This is most likely a factor of the large variation observed.