Dose-Time Response Modeling of Rat Brain AChE Activity: Formetanate Gavage Dosing

June 27, 2005

1 Preamble

Here is some code to set up the analysis: loading required libraries and datasets, and defining some functions.

First, `CarbamateData` loads the full dataset for this risk assessment, and causes the library `DRUtils` to be loaded.

```
> library(CarbamateData)

Loading required package: DRUtils
Loading required package: nlme
Loading required package: tcltk
```

Set up lattice to use B&W instead of color:

```
> library(lattice)
> ltheme <- canonical.theme(color = FALSE)
> ltheme$strip.background$col <- "transparent"
> lattice.options(default.theme = ltheme)
```

Use package `Hmisc` for some formatting support.

```
> library(Hmisc)

Hmisc library by Frank E Harrell Jr

Type library(help='Hmisc'), ?Overview, or ?Hmisc.Overview')
to see overall documentation.

NOTE:Hmisc no longer redefines [.factor to drop unused levels when
subsetting.  To get the old behavior of Hmisc type dropUnusedLevels().

Attaching package 'Hmisc':


        The following object(s) are masked from package:stats :

         ecdf
```

Only one submitted rat gavage study has brain measurements, but not for whole or half brain:

```
> unique(subset(AggData, chemical == "FORMETANATE" & species ==
+     "RAT" & dsmtd == "GAVAGE", select = c("mrid", "sctn")))
```

```
[1] mrid sctn
<0 rows> (or 0-length row.names)
```

This leaves just the Padilla dataset for formetanate. The following code prints out documentation for the datasets in use:

```
> printDataDoc(PadillaData)

----------------------------------------
Data set: PadillaData

Dataset creation date: Wed Jun 22 12:07:28 2005
Script name: /home/setzer/tasks/CarbamateCRA/EmpiricalDoseResponsesFinal/Data/getdata.R
Script last modified: 2005-06-22 12:05:47
sysname: Linux
release: 2.4.24-openmosix2
version: #1 Wed Jul 28 14:33:52 CEST 2004
nodename: node24.cluster2.org
machine: i686
login: unknown
user: setzer
----------------------------------------
```

The following function turns out to be quite useful on subsetted dataframes. It just eliminates unused levels of all factors in the data frame:

```
> CleanUp <- function(x) {
+     for (nm in names(x)) {
+         if (is.factor(x[, nm]))
+             x[, nm] <- factor(x[, nm])
+     }
+     x
+ }
```

To get starting values, we often have to extract values from a previously fit model. The following function simplifies that. The argument `what` is a regular expression:

```
> getParms <- function(what, Par) {
+     Par[grep(what, names(Par))]
+ }
```

This script is for modeling the dose-time response for rat red blood cells. via gavage dosing. It includes two acute studies.

Now, set up the analysis dataset.

```
> dta <- subset(PadillaData, chemical %in% "formetanate", select = c("dose",
+     "brain.R", "TMPSTDS"))
> names(dta) <- c("dose", "cheact", "tmpstds")
> dta <- dta[!is.na(dta$cheact), ]
```

Summary of the relevant variables in this dataset:

```
> summary(dta)

      dose            cheact        tmpstds
 Min.   : 0.000   Min.   : 527   Min.   : 0.5000
 1st Qu.: 0.250   1st Qu.:1853   1st Qu.: 0.6667
```

```
 Median : 3.250   Median :4203   Median : 0.6667
 Mean   : 4.792   Mean   :4317   Mean   : 3.4833
 3rd Qu.:10.000   3rd Qu.:6635   3rd Qu.: 2.0000
 Max.   :10.000   Max.   :8029   Max.   :24.0000
```

```
> unique(subset(dta, select = c("dose", "tmpstds")))
```

```
     dose    tmpstds
156   0.0  0.6666667
161   0.1  0.6666667
166   0.3  0.6666667
171   0.6  0.6666667
176   1.5  0.6666667
181   5.0  0.6666667
1571  0.0  0.5000000
1581  0.0  1.0000000
1591  0.0  2.0000000
1601  0.0 24.0000000
1611  0.0  4.0000000
1621 10.0  0.5000000
1671 10.0  1.0000000
1721 10.0  2.0000000
1771 10.0  4.0000000
1821 10.0 24.0000000
```

Padilla's data are from an acute study, with multiple doses at 0.67 hour, and multiple timepoints for 10 mg/kg.

Set up a factor so that I can have a different control value for each separate time:

```
> dta$ftmpstds <- factor(dta$tmpstds)
```

2 Dose-Response Modeling

2.1 strategy

Use the model with simple exponential recovery (`tcmfn4()`), first. It looks as if the time to peak effect for all these chemicals is likely to be less than a half-hour (the earliest time point in this dataset), so the exponential recovery model is probably indistinguishable from the one with the more complex time course.

Fitting the model will follow these steps:

1. First, use `GetInitialValues()` to get starting values for the model against these data, and determine whether we can estimate `tz` and/or `lg`.

2. Next, fit `tcmfn4()` using the parameterizations determined in the previous step, and a constant variance model.

   Try fitting with other variance models the same model for fixed and random effects, using AIC to identify the best variance model

There are no random effects in this model, so it can be fit using `gnls()`.

2.2 Initial Values

Save the initial values so that we do not need to go through all this to re-run the analysis. Also, set the argument `delta` to 0.5, the earliest non-zero time point.

```
> formals(tcmfn4)$delta <- min(dta$tmpstds[dta$tmpstds > 0])
> initfile <- paste("initvals-Brain-DR-1.RData", sep = "")
> if (!file.exists(initfile)) {
+     lA.start <- lm(I(log(cheact)) ~ ftmpstds - 1, data = CleanUp(subset(dta,
+         dose %in% 0)))
+     Start <- c(coef(lA.start), log(0.3), -2, log(1), log(1.5))
+     init1 <- GetInitialValues(cheact ~ tcmfn4(dose, tmpstds,
+         lA = lA, tz = tz, lD = lD, lg = lg, lTr = lTr), data = dta,
+         params = list(lA ~ ftmpstds - 1, lD ~ 1, tz ~ 1, lg ~
+             1, lTr ~ 1), start = Start, weights = varPower(value = 1))
+     save(init1, file = initfile)
+ } else load(initfile)
> tmp <- t(init1$Redundancy[[1]]$Eigens)
> tmp <- tmp[-grep("^lA", rownames(tmp)), ]
> round(tmp[, 1:7], digits = 2)

           [,1] [,2] [,3] [,4] [,5] [,6] [,7]
CondIndex 25.68 7.15 4.01 2.54 1.87 1.87 1.87
mu         0.07 0.26 0.47 0.74 1.00 1.00 1.00
lD         0.93 0.07 0.00 0.00 0.00 0.00 0.00
tz         0.59 0.40 0.00 0.01 0.00 0.00 0.00
lg         0.99 0.00 0.00 0.00 0.00 0.00 0.00
lTr        0.00 0.02 0.89 0.00 0.00 0.00 0.00
```

This might work.

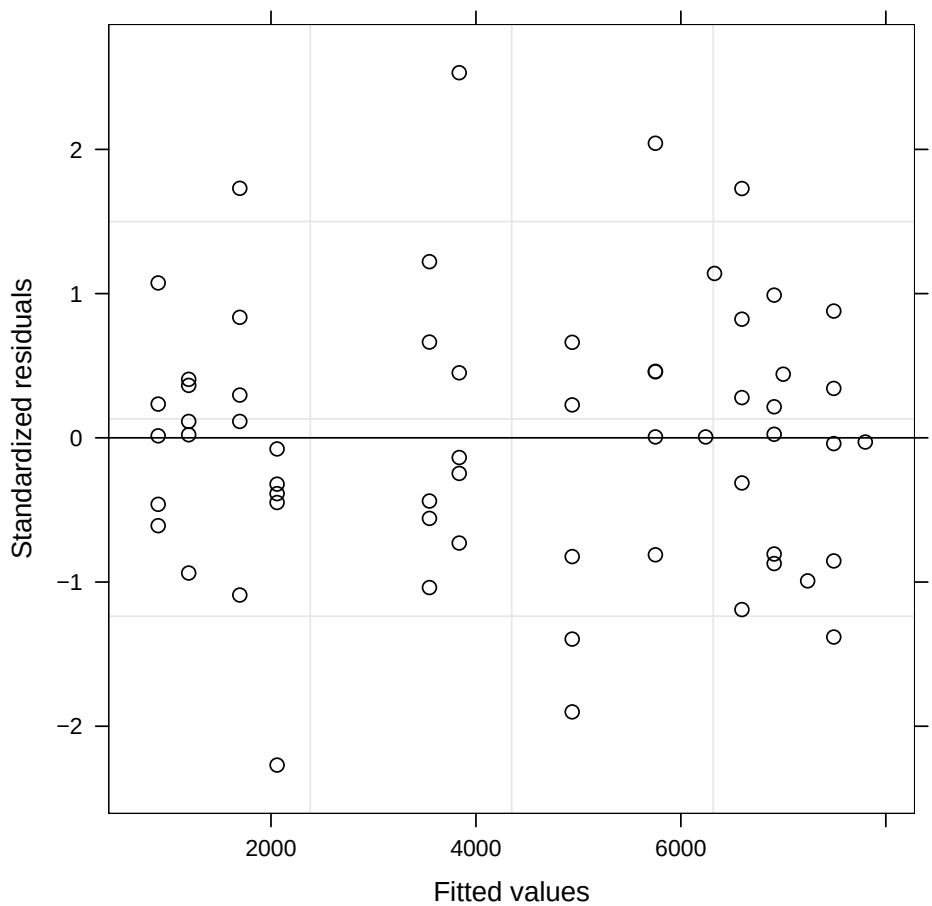
```
> Start <- Start0 <- init1$start$beta
> icnt <- 1
> Maxcnt <- 50
> repeat {
+     drmod1 <- try(gnls(cheact ~ tcmfn4(dose, tmpstds, lA = lA,
+         tz = tz, lD = lD, lg = lg, lTr = lTr), data = dta, params = list(lA ~
+         ftmpstds - 1, tz ~ 1, lD ~ 1, lg ~ 1, lTr ~ 1), weights = varPower(value = 1),
+         start = Start), silent = TRUE)
+     if (!inherits(drmod1, "try-error") || icnt >= Maxcnt)
+         break
+     else {
+         icnt <- icnt + 1
+         Start <- Start0 * (1 + rnorm(length(Start0), sd = 0.2))
+     }
+ }
> if (icnt < Maxcnt) writeLines(paste("Success in", icnt, "tries")) else writeLines("Maxcnt exceeded")

Success in 22 tries
```

Now, do we need to model the variance?

```
> Start <- Start0 <- coef(drmod1)
> icnt <- 1
> Maxcnt <- 50
> repeat {
+     drmod2 <- try(gnls(cheact ~ tcmfn4(dose, tmpstds, lA = lA,
+         tz = tz, lD = lD, lg = lg, lTr = lTr), data = dta, params = list(lA ~
+         ftmpstds - 1, tz ~ 1, lD ~ 1, lg ~ 1, lTr ~ 1), start = Start),
+         silent = TRUE)
```

```
+     if (!inherits(drmod2, "try-error") || icnt >= Maxcnt)
+         break
+     else {
+         icnt <- icnt + 1
+         Start <- Start0 * (1 + rnorm(length(Start0), sd = 0.2))
+     }
+ }
> if (icnt < Maxcnt) writeLines(paste("Success in", icnt, "tries")) else writeLines("Maxcnt exceeded")

Success in 5 tries

> anova(drmod2, drmod1)

       Model df      AIC      BIC    logLik   Test   L.Ratio p-value
drmod2     1 11 951.1420 974.1797 -464.5710
drmod1     2 12 952.2398 977.3719 -464.1199 1 vs 2 0.9021422  0.3422
```

No. Plot the second model:



Now, try plotting versus fraction of inhibition:

```
> local({
+     tmp <- drmod2
+     tmp$contrasts <- NULL
```

```
+     ndta <- dta
+     ndta$dose[] <- 0
+     BG <- predict(tmp, newdata = ndta)
+     Inh <- 1 - fitted(drmod2)/BG
+     plot(resid(drmod2, type = "p") ~ Inh, xlab = "Fractional Inhibition",
+         ylab = "Pearson Residual")
+ })
```

That looks pretty good.

The interesting fitted values:

```
> nms <- names(coef(drmod2))
> summary(drmod2)$tTable[-grep("^lA", nms), ]

         Value Std.Error   t-value      p-value
tz  -2.3726398 0.9012564 -2.632591 1.124256e-02
lD  -2.6349501 0.4629463 -5.691697 6.570326e-07
lg  -0.3963653 0.1593358 -2.487609 1.623993e-02
lTr  1.3436572 0.1837494  7.312444 1.949079e-09
```

Now, finally, are the background levels different?

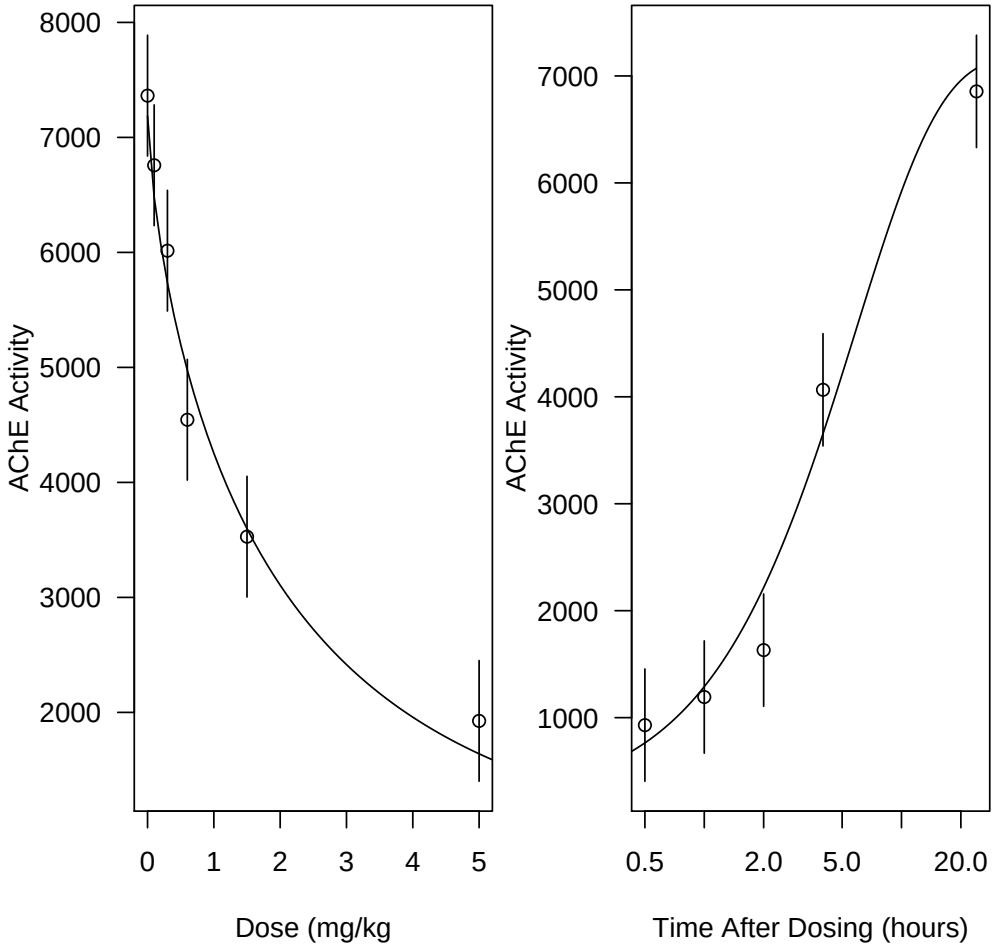
```
> lapars <- getParms("^lA", coef(drmod2))
> ncont <- length(lapars)
> L <- diag(ncont) - matrix(1, nrow = ncont, ncol = ncont)/ncont
> L <- L[-ncont, ]
> dimnames(L) <- list(NULL, names(lapars))
> anova(drmod2, L = L)

Denom. DF: 50
 F-test for linear combination(s)
  lA.ftmpstds0.5 lA.ftmpstds0.666666666666667 lA.ftmpstds1 lA.ftmpstds2
1      0.8333333                   -0.1666667   -0.1666667   -0.1666667
2     -0.1666667                    0.8333333   -0.1666667   -0.1666667
3     -0.1666667                   -0.1666667    0.8333333   -0.1666667
4     -0.1666667                   -0.1666667   -0.1666667    0.8333333
5     -0.1666667                   -0.1666667   -0.1666667   -0.1666667
  lA.ftmpstds4 lA.ftmpstds24
1   -0.1666667    -0.1666667
2   -0.1666667    -0.1666667
3   -0.1666667    -0.1666667
4   -0.1666667    -0.1666667
5    0.8333333    -0.1666667
  numDF  F-value p-value
1     5 1.591447  0.1797
```

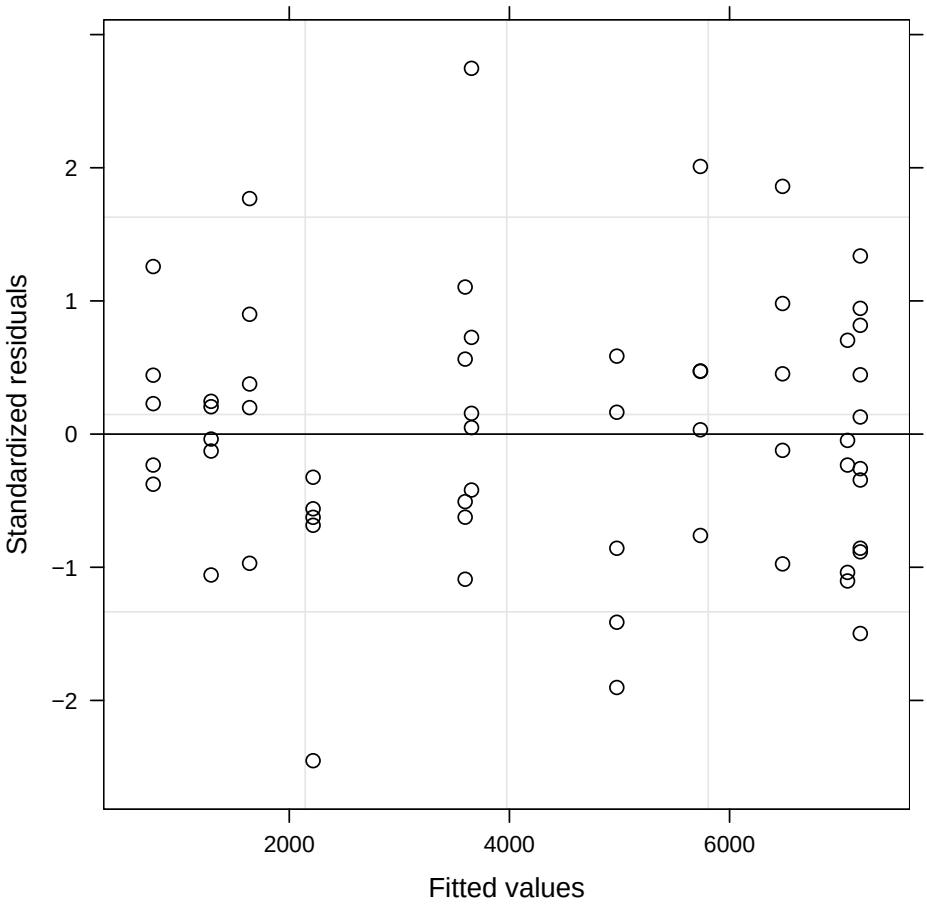
No, so refit the data with a simpler background term:

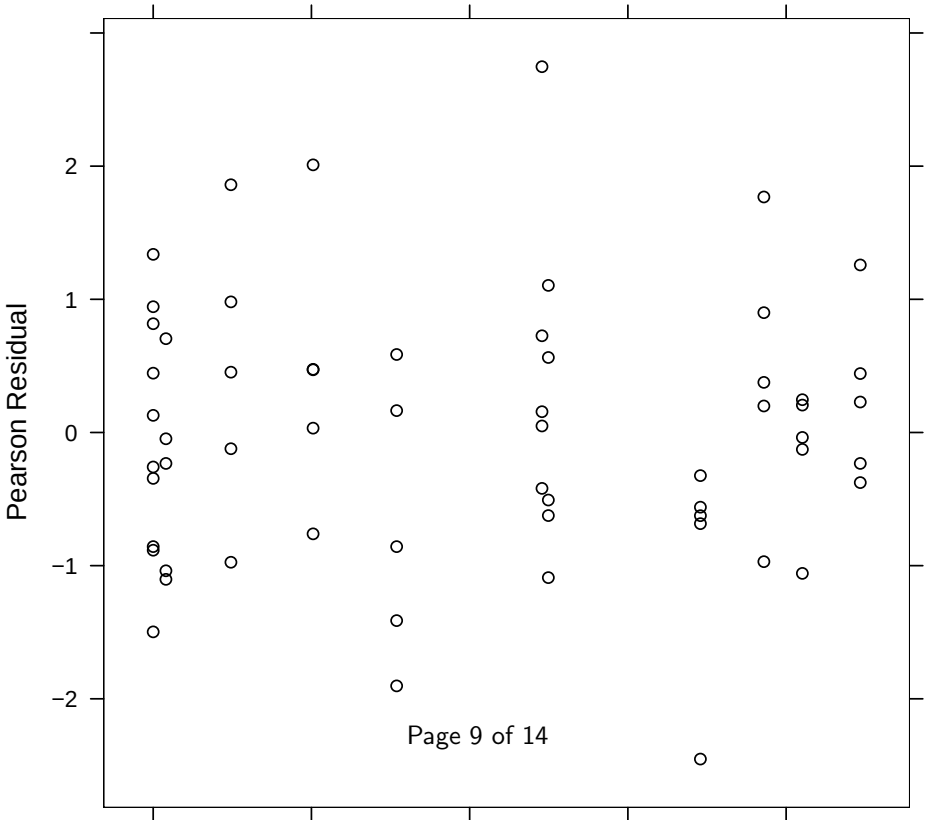
```
> Par <- coef(drmod2)
> Start <- Start0 <- c(mean(getParms("^lA", Par)), Par["tz"], Par["lD"],
+     Par["lg"], Par["lTr"])
> icnt <- 1
> Maxcnt <- 60
> repeat {
+     drmod3 <- try(gnls(cheact ~ tcmfn4(dose, tmpstds, lA = lA,
+         tz = tz, lD = lD, lg = lg, lTr = lTr), data = dta, params = list(lA ~
```

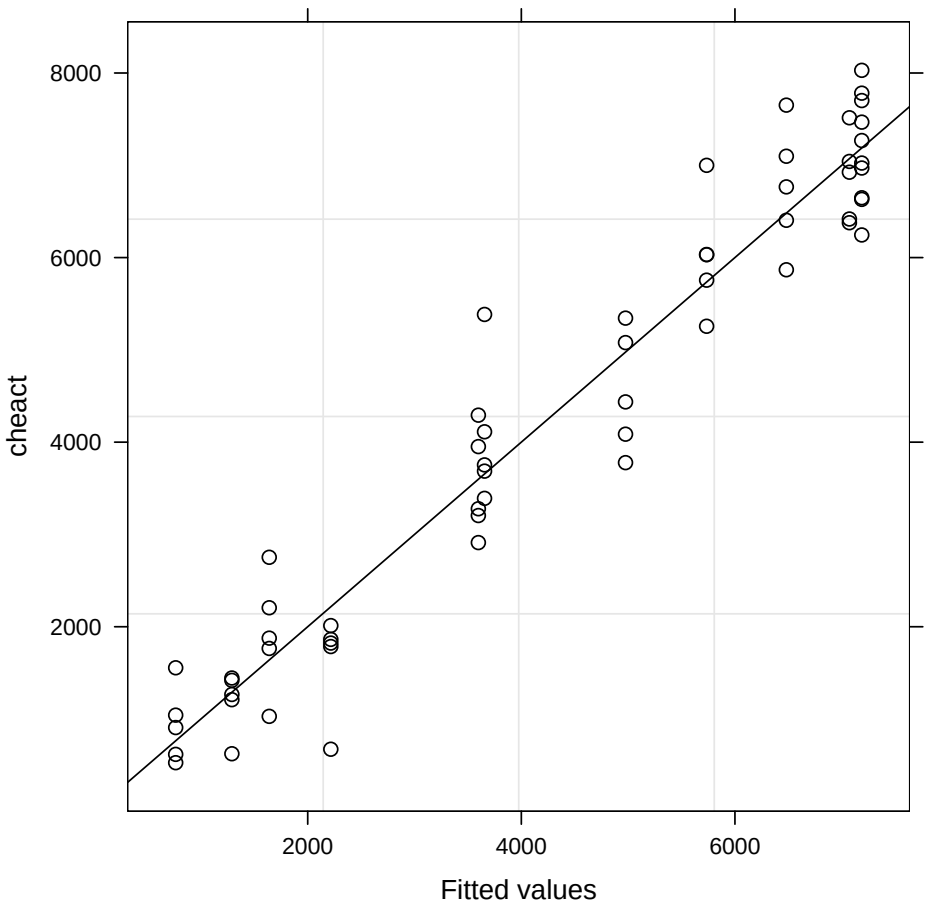
```
+         1, tz ~ 1, lD ~ 1, lg ~ 1, lTr ~ 1), start = Start),
+         silent = TRUE)
+     if (!inherits(drmod3, "try-error") || icnt > Maxcnt) {
+         if (icnt <= Maxcnt)
+             writeLines(paste("Success after", icnt, "trie(s)."))
+         else writeLines(paste("Maxcnt,", Maxcnt, "exceeded"))
+         break
+     }
+     Start <- Start0 * (1 + rnorm(length(Start0), sd = 0.2))
+     icnt <- icnt + 1
+ }

Success after 28 trie(s).

> Ints <- intervals(drmod3, which = "coef")$coef
> Ints90 <- intervals(drmod3, which = "coef", level = 0.9)$coef
> tTab <- summary(drmod3)$tTable
> summary(drmod3)

Generalized nonlinear least squares fit
  Model: cheact ~ tcmfn4(dose, tmpstds, lA = lA, tz = tz, lD = lD, lg = lg,      lTr = lTr)
  Data: dta
       AIC      BIC    logLik
  950.3996 962.9657 -469.1998

Coefficients:
        Value Std.Error  t-value p-value
lA   8.880087 0.0230376 385.4608  0.0000
tz  -2.565145 1.1075977  -2.3160  0.0243
lD  -2.318631 0.3849400  -6.0234  0.0000
lg  -0.332836 0.1538444  -2.1635  0.0349
lTr  1.400183 0.1465676   9.5532  0.0000

 Correlation:
    lA     tz     lD     lg
tz  -0.146
lD  -0.554  0.577
lg  -0.399  0.773  0.928
lTr  0.326  0.374 -0.095  0.002

Standardized residuals:
         Min           Q1          Med           Q3          Max
-2.452429418 -0.639721013 -0.002313827  0.496483218  2.746880828

Residual standard error: 629.272
Degrees of freedom: 60 total; 55 residual
```

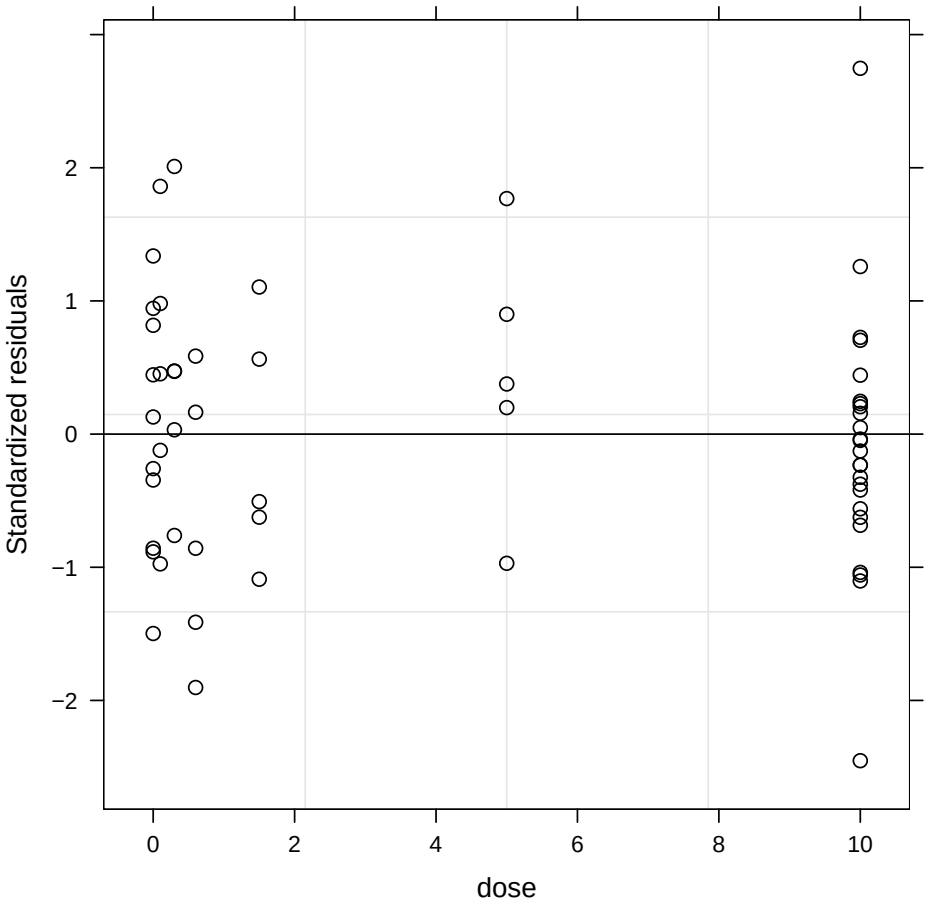
Diagnostic plots for this model: First, the dose-response and recovery curves with means and 95% confidence intervals from the data.

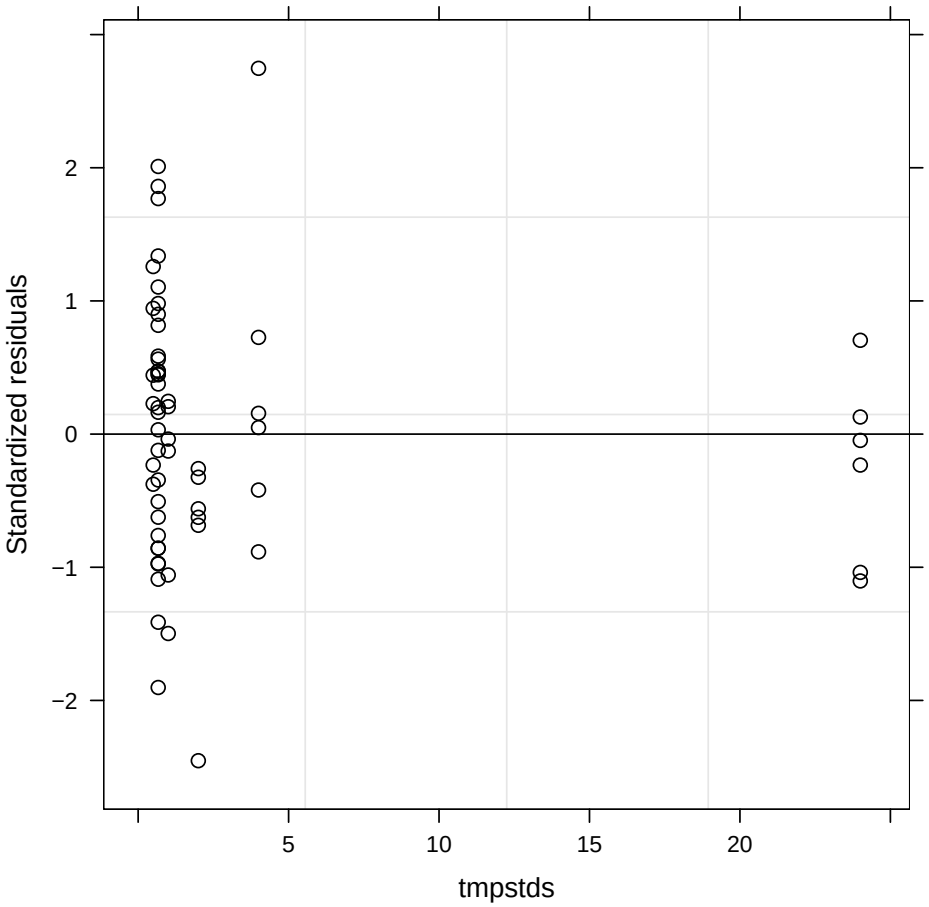

8000
7000
6000
5000
4000
3000
2000
AChE Activity
0
1
2
3
4
5
Dose (mg/kg
7000
6000
5000
4000
3000
2000
1000
AChE Activity
0.5
2.0
5.0
20.0
Time After Dosing (hours)

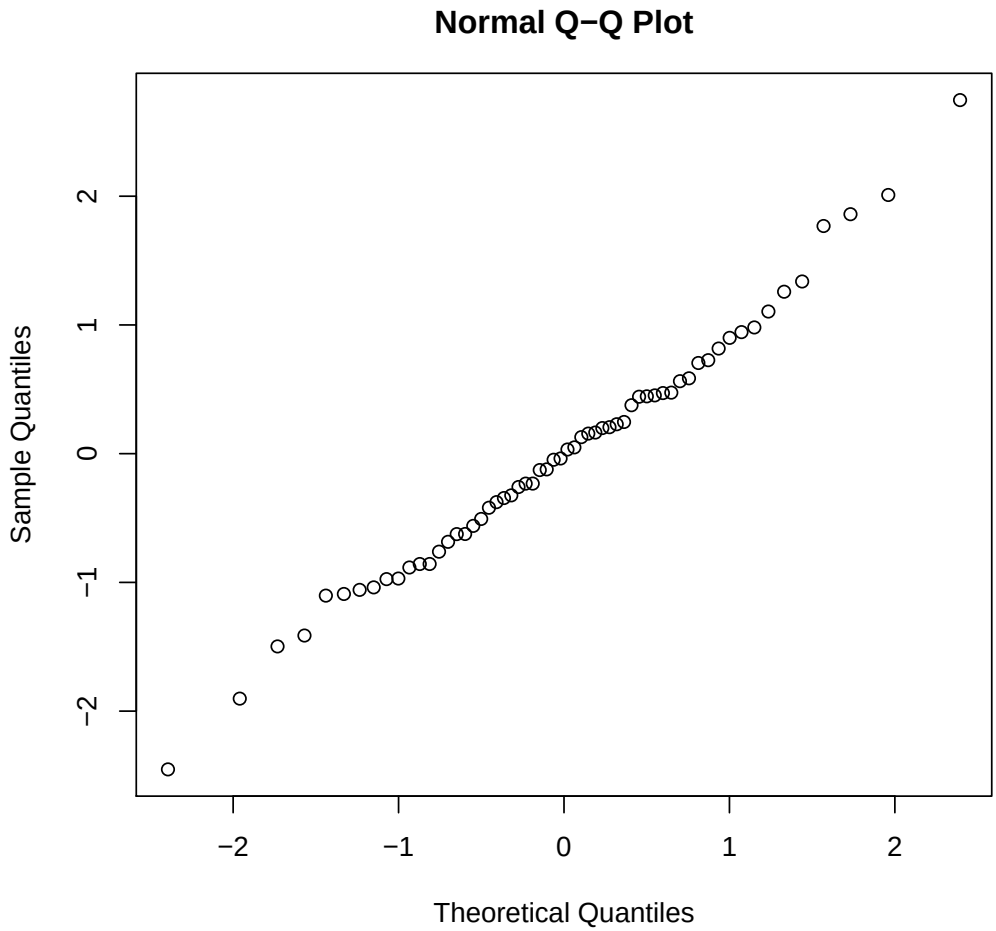

Standardized residuals
Fitted values
2
1
0
−1
−2
2000
4000
6000


Pearson Residual
2
1
0
−1
−2


8000
6000
4000
2000
cheact
2000
4000
6000
Fitted values


Standardized residuals
2
1
0
−1
−2
0
2
4
6
8
10
dose


Standardized residuals
2
1
0
−1
−2
5
10
15
20
tmpstds



3 Summary

The critical estimates from this analysis are listed below. They are printed with greater than usual precision, in case they are to be used in further computation. For reporting, round to two or three significant digits. BMD has units mg/kg, and times are in hours.

species RAT

mrid PADILLA

`lD` **(se)** -2.31863131185074 (0.384939956594312)

BMD (95% CI) 0.0984081835866834 (0.0454988501968644, 0.21284429287617)

BMDL, the one-sided lower 95% CL 0.0516818339323545

`lTr (standard error)` 1.40018251875842 (0.146567576432641)

Recovery Half-life (95% CI) 4.0559401844575 (3.02361922221520, 5.44071510692567)

Save the results:

```
> save.image(file = "RatBrainDR.RData")
```

Save the results for incorporating into a database:

```
> formetenate.oral.brain <- list(mrid = levels(dta$mrid), species = "RAT",
+     BMDs = list(male = list(lD = tTab["lD", "Value"], lD.se = tTab["lD",
+         "Std.Error"], BMD = exp(Ints["lD", "est."]), BMD.CI = exp(Ints["lD",
+         c("lower", "upper")]), BMDL = exp(Ints90["lD", "lower"]))),
+     HalfLives = list(lTr = tTab["lTr", "Value"], lTr.se = tTab["lTr",
+         "Std.Error"], Tr = exp(Ints["lTr", "est."]), Tr.CI = exp(Ints["lTr",
+         c("lower", "upper")])))
> save(formetenate.oral.brain, file = file.path("..", "..", "01Summaries",
+     "formetenate.oral.brain.RData"))
```