Dose-Time Response Modeling of Rat RBC 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
```

There are two submitted rat gavage studies, but they both just measured whole blood:

```
> 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",
+     "RBC.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.   : 21.77   Min.   : 0.5000
 1st Qu.: 0.200   1st Qu.:106.14   1st Qu.: 0.6667
```

```
 Median : 1.500   Median :216.23   Median : 0.6667
 Mean   : 4.703   Mean   :230.17   Mean   : 3.5085
 3rd Qu.:10.000   3rd Qu.:347.13   3rd Qu.: 2.0000
 Max.   :10.000   Max.   :544.43   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
1731 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$tmpstds > 0])
> initfile <- paste("initvals-RBC-DR-1.RData", sep = "")
> if (!file.exists(initfile)) {
+     lA.start <- lm(I(log(cheact)) ~ ftmpstds - 1, data = CleanUp(subset(dta,
+         dose %in% 0)))
+     Start <- c(coef(lA.start), log(0.3), -2, log(1), log(1.5))
+     init1 <- GetInitialValues(cheact ~ tcmfn4(dose, tmpstds,
+         lA = lA, tz = tz, lD = lD, lg = lg, lTr = lTr), data = dta,
+         params = list(lA ~ ftmpstds - 1, lD ~ 1, tz ~ 1, lg ~
+             1, lTr ~ 1), start = Start, weights = varPower(value = 1))
+     save(init1, file = initfile)
+ } else load(initfile)
> tmp <- t(init1$Redundancy[[1]]$Eigens)
> tmp <- tmp[-grep("^lA", rownames(tmp)), ]
> round(tmp[, 1:7], digits = 2)
```

```
           [,1] [,2] [,3] [,4] [,5] [,6] [,7]
CondIndex 31.44 8.73 4.16 2.60 1.89 1.89 1.89
mu         0.06 0.22 0.45 0.73 1.00 1.00 1.00
lD         0.92 0.07 0.00 0.00 0.00 0.00 0.00
tz         0.61 0.38 0.00 0.00 0.00 0.00 0.00
lg         1.00 0.00 0.00 0.00 0.00 0.00 0.00
lTr        0.01 0.04 0.87 0.00 0.00 0.00 0.00
```

The largest condition index is over 31, as usual, with fairly strong redundancy among the dose-response parameters. More importantly, optimizing over the model above pushes `tz` to very small values. Set `tz` to -10, and don't estimate it.

```
> Start <- init1$start$beta
> Start <- Start0 <- Start[-grep("tz", names(Start))]
> formals(tcmfn4)$tz <- -10
> icnt <- 1
> Maxcnt <- 20
> repeat {
+     drmod1 <- try(gnls(cheact ~ tcmfn4(dose, tmpstds, lA = lA,
+         lD = lD, lg = lg, lTr = lTr), data = dta, params = list(lA ~
+         ftmpstds - 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")
```

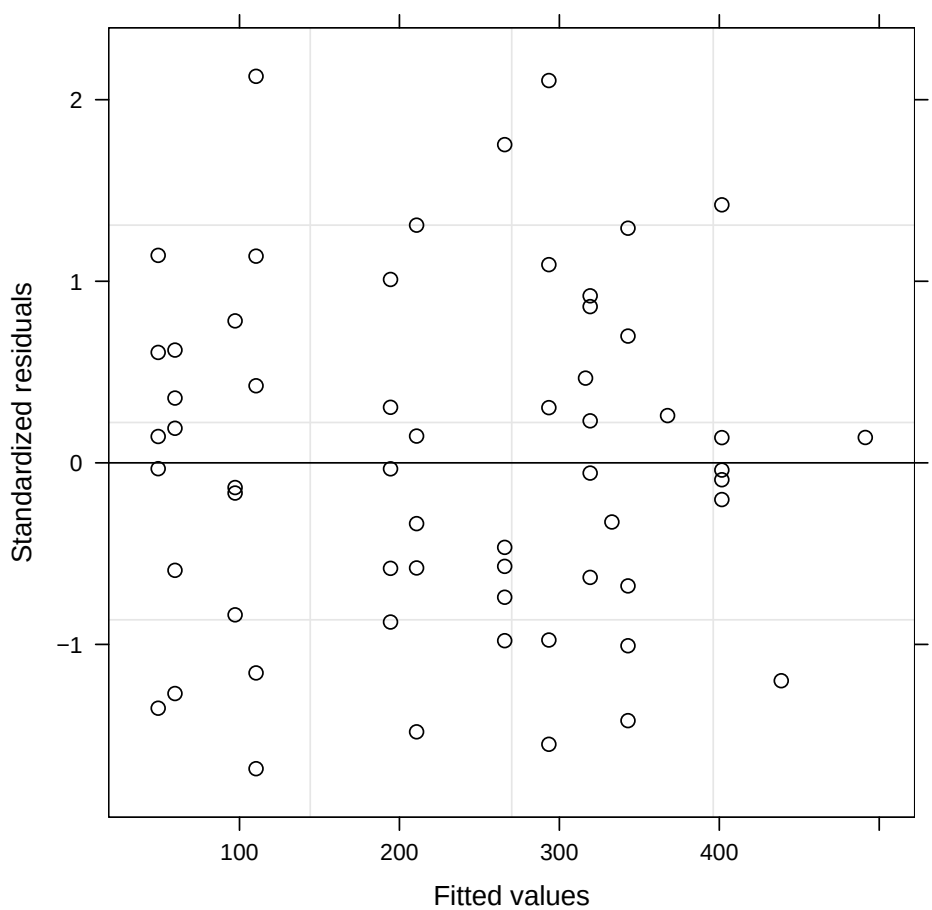
```
Success in 1 tries
```

Now, do we need to model the variance?

```
> Start <- Start0 <- coef(drmod1)
> icnt <- 1
> Maxcnt <- 20
> repeat {
```

```
+     drmod2 <- try(gnls(cheact ~ tcmfn4(dose, tmpstds, lA = lA,
+         lD = lD, lg = lg, lTr = lTr), data = dta, params = list(lA ~
+         ftmpstds - 1, lD ~ 1, lg ~ 1, lTr ~ 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 1 tries

> anova(drmod2, drmod1)

       Model df      AIC      BIC    logLik   Test  L.Ratio p-value
drmod2     1 10 673.4714 694.2468 -326.7357
drmod1     2 11 655.3486 678.2015 -316.6743 1 vs 2 20.12278  <.0001
```

Yes. Plot the model, to see if there looks to be any additional trend of residual magnitude versus fitted value:



Now, try plotting versus fraction of inhibition:

```
> local({
+     tmp <- drmod1
+     tmp$contrasts <- NULL
+     ndta <- dta
+     ndta$dose[] <- 0
+     BG <- predict(tmp, newdata = ndta)
+     Inh <- 1 - fitted(drmod1)/BG
+     plot(resid(drmod1, type = "p") ~ Inh, xlab = "Fractional Inhibition",
+         ylab = "Pearson Residual")
+ })
```

That looks pretty good.

The interesting fitted values:

```
> nms <- names(coef(drmod1))
> summary(drmod1)$tTable[-grep("^lA", nms), ]

         Value Std.Error   t-value      p-value
lD  -1.7800306 0.9916700 -1.794983 7.870207e-02
lg  -0.3423833 0.2583518 -1.325260 1.911090e-01
lTr  1.9535136 0.4417668  4.422047 5.280348e-05
```

Now, finally, are the background levels different?

```
> lapars <- getParms("^lA", coef(drmod1))
> ncont <- length(lapars)
> L <- diag(ncont) - matrix(1, nrow = ncont, ncol = ncont)/ncont
> L <- L[-ncont, ]
> dimnames(L) <- list(NULL, names(lapars))
> anova(drmod1, 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 0.9992804  0.4279
```

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

```
> Par <- coef(drmod1)
> Start <- Start0 <- c(mean(getParms("^lA", Par)), Par["lD"], Par["lg"],
+     Par["lTr"])
> icnt <- 1
> Maxcnt <- 30
> repeat {
```

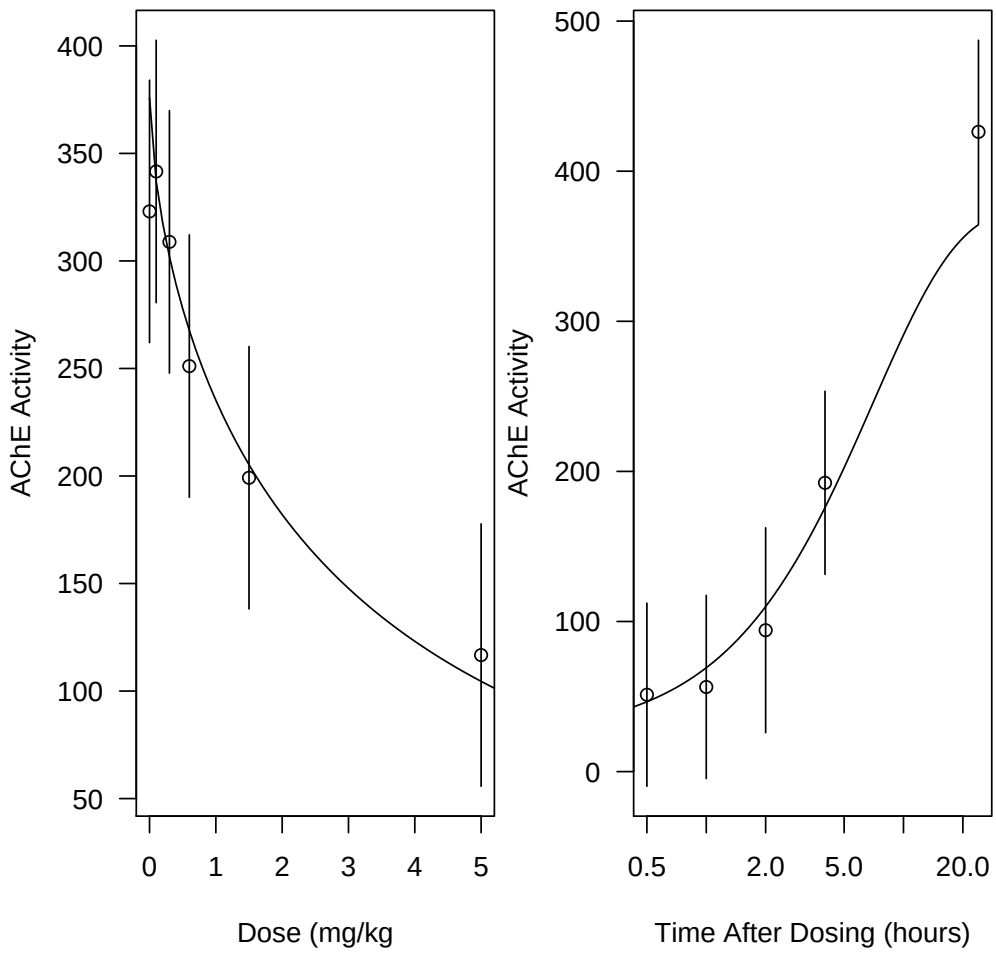
```
+     drmod3 <- try(gnls(cheact ~ tcmfn4(dose, tmpstds, lA = lA,
+         lD = lD, lg = lg, lTr = lTr), data = dta, params = list(lA ~
+         1, lD ~ 1, lg ~ 1, lTr ~ 1), weights = varPower(value = 1),
+         start = Start), silent = TRUE)
+     if (!inherits(drmod3, "try-error") || icnt > 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
+ }
```

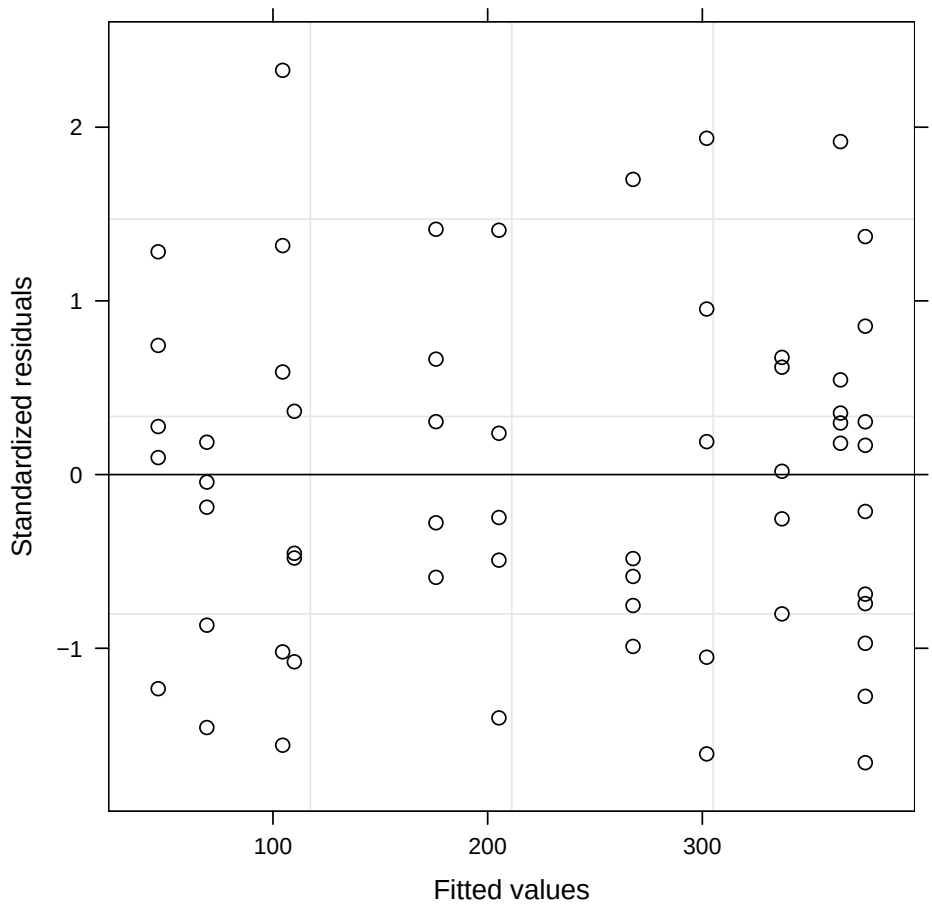
```
Success after 1 trie(s).
```

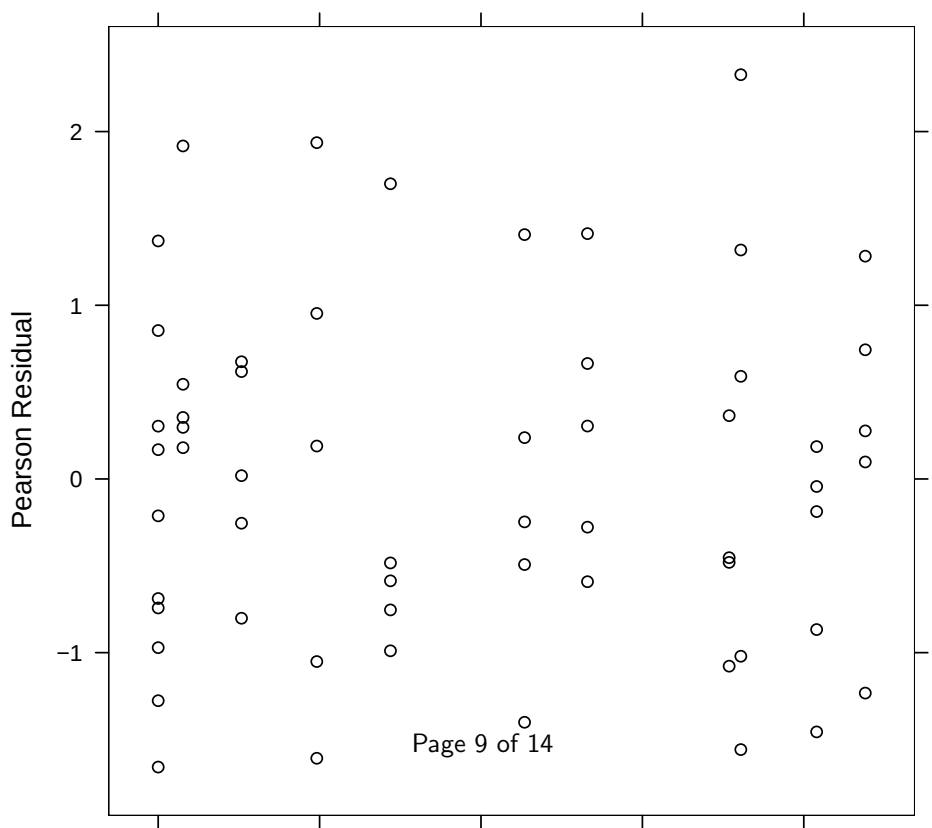
```
> Ints <- intervals(drmod3, which = "coef")$coef
> Ints90 <- intervals(drmod3, which = "coef", level = 0.9)$coef
> tTab <- summary(drmod3)$tTable
> summary(drmod3)
```

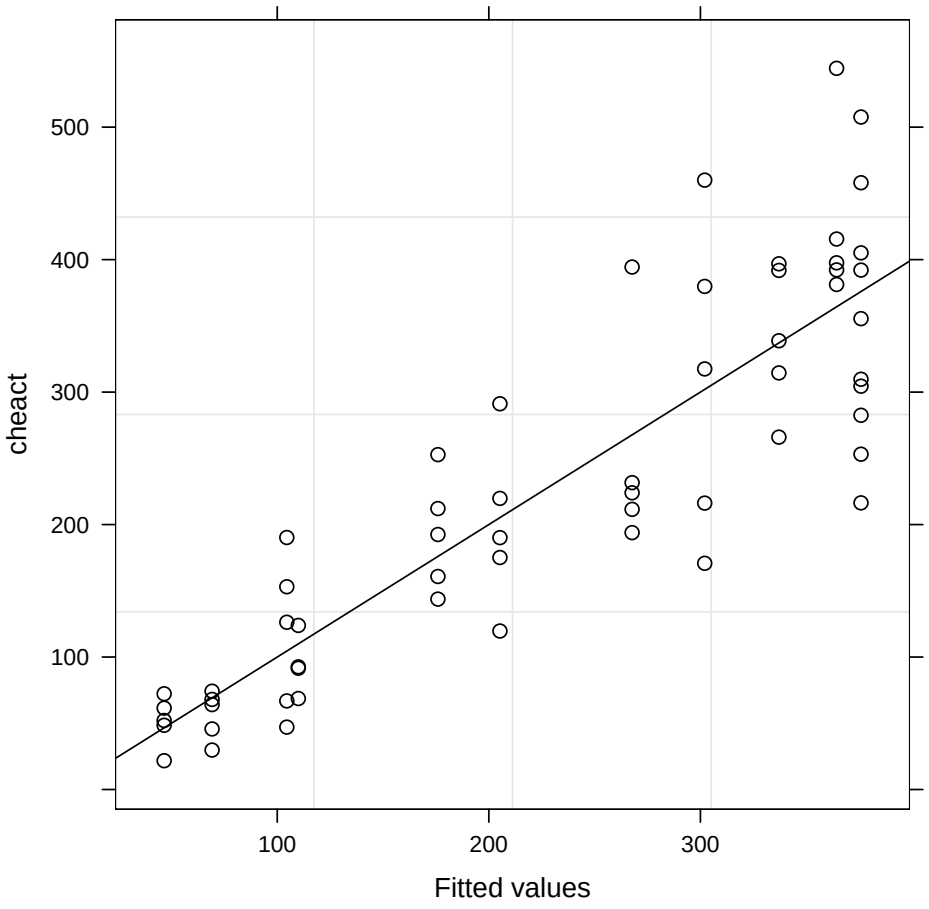
```
Generalized nonlinear least squares fit
  Model: cheact ~ tcmfn4(dose, tmpstds, lA = lA, lD = lD, lg = lg, lTr = lTr)
  Data: dta
       AIC      BIC    logLik
  653.3085 665.7737 -320.6543

Variance function:
 Structure: Power of variance covariate
 Formula: ~fitted(.)
 Parameter estimates:
    power
0.7494814

Coefficients:
        Value Std.Error  t-value p-value
lA   5.929309 0.0652303 90.89812  0.0000
lD  -2.395144 0.7147699 -3.35093  0.0015
lg  -0.452755 0.1555340 -2.91097  0.0052
lTr  1.581382 0.2364981  6.68666  0.0000

 Correlation:
    lA     lD     lg
lD  -0.676
lg  -0.619  0.986
lTr  0.494 -0.395 -0.429

Standardized residuals:
        Min          Q1         Med          Q3         Max
-1.65875473 -0.74825931  0.01913042  0.60461878  2.32757886

Residual standard error: 1.129780
Degrees of freedom: 59 total; 55 residual
```

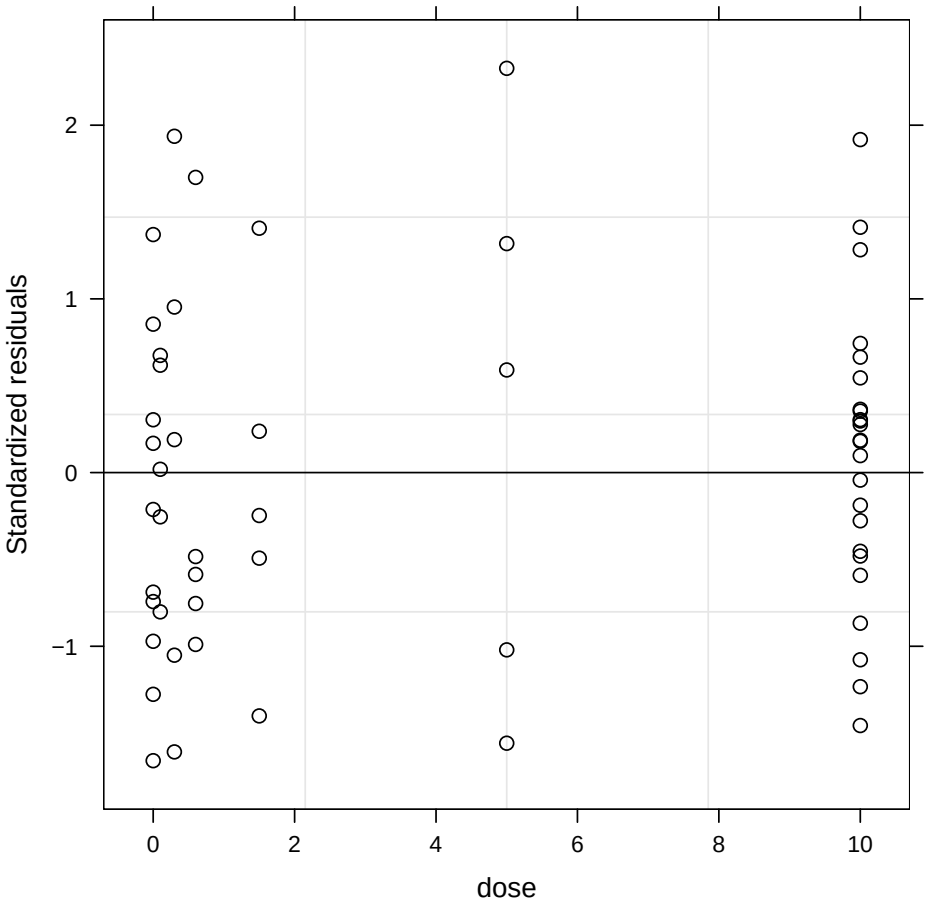
Diagnostic plots for this model: First, the dose-response and recovery curves with means and 95% confidence intervals from the data.

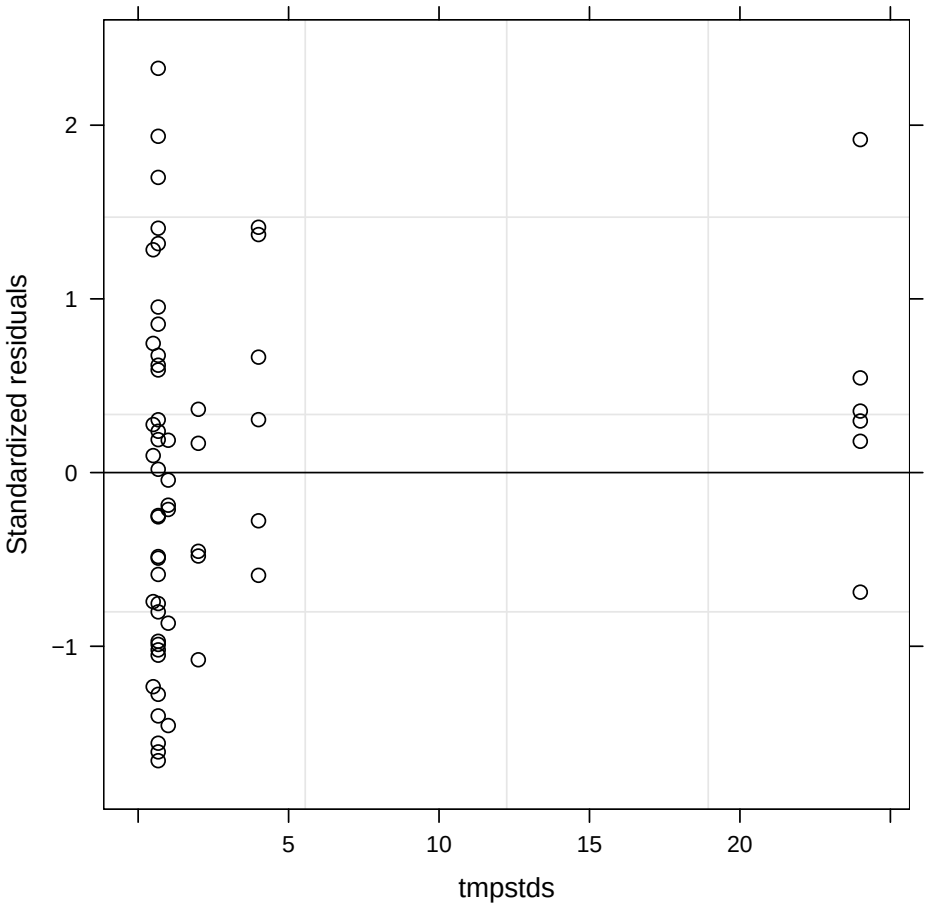

AChE Activity
400
350
300
250
200
150
100
50
0
1
2
3
4
5
Dose (mg/kg
AChE Activity
500
400
300
200
100
0
0.5
2.0
5.0
20.0
Time After Dosing (hours)

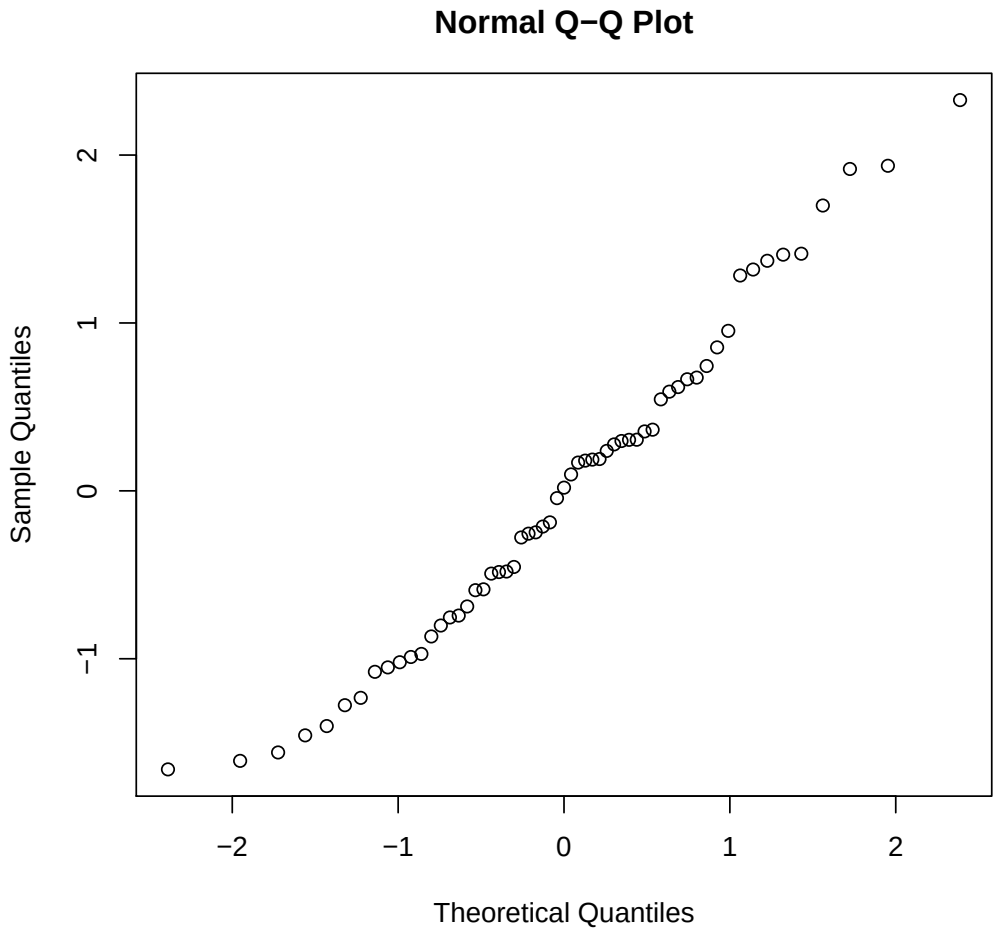

Standardized residuals
Fitted values
2
1
0
−1
100
200
300


Pearson Residual
2
1
0
−1


cheact
500
400
300
200
100
100
200
300
Fitted values


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


Standardized residuals
2
1
0
−1
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)** -2.39514396116007 (0.714769885578618)

BMD (95% CI) 0.0911595545426963 (0.0217623292147017, 0.38185546696026)

BMDL, the one-sided lower 95% CL 0.0275713453618948

`lTr (standard error)` 1.58138200423247 (0.236498063506611)

Recovery Half-life (95% CI) 4.86167001918026 (3.0265679682491, 7.80945137309102)

Save the results:

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

Save the results for incorporating into a database:

```
> formetenate.oral.rbc <- 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.rbc, file = file.path("..", "..", "01Summaries",
+     "formetenate.oral.rbc.RData"))
```