Dose-Time Response Modeling of Rat Brain AChE Activity: Carbofuran 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
```

All the rat gavage data for this analysis are in `PadillaData`. 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 brain. via gavage dosing. It includes only an acute study.

All the data used for this DR model are in `PadillaData`. Now, set up the analysis dataset.

```
> dta <- subset(PadillaData, chemical %in% "carbofuran", select = c("dose",
+     "brain.R", "TMPSTDS"))
> names(dta) <- c("dose", "cheact", "tmpstds")
```

Summary of this dataset:

```
> summary(dta)

      dose            cheact        tmpstds
 Min.   :0.0000   Min.   :1957   Min.   : 0.5000
 1st Qu.:0.3000   1st Qu.:4610   1st Qu.: 0.6667
 Median :0.5000   Median :5806   Median : 0.6667
 Mean   :0.4654   Mean   :5656   Mean   : 3.4000
 3rd Qu.:0.5000   3rd Qu.:6777   3rd Qu.: 4.0000
 Max.   :1.5000   Max.   :7796   Max.   :24.0000
```

Padilla's data are from an acute study, with multiple doses at one time point (2/3 hr), and multiple timepoints for 0.5 mg/kg.

There is a control for each time point, so we need to set up a factor that captures those levels.

```
> dta$ftime <- factor(dta$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 `lg` and `tz` of the dose-response parameters.
2. Next, fit `tcmfn4()` using the parameterizations determined in the previous step. Since this is a single data set, with no repeated measures, use `gnls()`.
3. Try fitting with other variance models the same model for fixed and random effects, using AIC to identify the best variance model. Once the best model for the error variance is found, we are done.

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$tmpstds > 0])
> initfile <- paste("initvals-brain-DR-1.RData", sep = "")
> if (!file.exists(initfile)) {
+     lA.start <- lm(I(log(cheact)) ~ ftime - 1, data = CleanUp(subset(dta,
+         dose %in% 0)))
+     Start <- c(coef(lA.start), log(0.1), -2, log(1), log(1.5))
+     init1 <- GetInitialValues(cheact ~ tcmfn4(dose, tmpstds,
+         lA = lA, tz = tz, lD = lD, lg = lg, lTr = lTr), data = dta,
+         params = list(lA ~ ftime - 1, lD ~ 1, tz ~ 1, lg ~ 1,
+             lTr ~ 1), start = Start, weights = varPower(value = 1))
+     save(init1, file = initfile)
+ } else load(initfile)
> tmp <- t(init1$Redundancy[[1]]$Eigens)
> tmp <- tmp[-grep("^lA", rownames(tmp)), ]
> round(tmp[, 1:5], digits = 2)
```

```
           [,1] [,2] [,3] [,4] [,5]
CondIndex 12.47 5.55 2.80 1.73 1.73
mu         0.14 0.31 0.62 1.00 1.00
lD         0.99 0.00 0.01 0.00 0.00
lg         0.95 0.02 0.03 0.00 0.00
lTr        0.04 0.92 0.00 0.00 0.00
```

The high-dose shape parameter `tz` ends up being set very small, where continued reduction yields no change in fit, so set it to -10 in the subsequent analysis. As usual, `lD` and `lg` are potentially redundant, but in this dataset the condition index of their eigenvector does not seem to be extremely large, so it may be possible to fit both parameters Try fitting the full model first, but if it fails, try refitting with a single value of `lg`.

```
> Parm <- init1$start$beta
> Start <- Start0 <- c(getParms("^lA", Parm), Parm["lD"], Parm["lg"],
+     Parm["lTr"])
> idta <- init1$data
> formals(tcmfn4)$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 ~
+         ftime - 1, lD ~ 1, lg ~ 1, lTr ~ 1), start = Start))
+     if (!inherits(drmod1, "try-error") || icnt > Maxcnt) {
+         if (icnt <= Maxcnt)
+             writeLines(paste("Successful in ", icnt, if (icnt >
+                 1)
+                 "tries"
+             else "try"))
+         else writeLines("Maxcnt exceeded")
+         break
+     }
+     icnt <- icnt + 1
+     Start <- Start0 * (1 + rnorm(length(Start0), sd = 0.2))
+ }
```

This fails with the error:

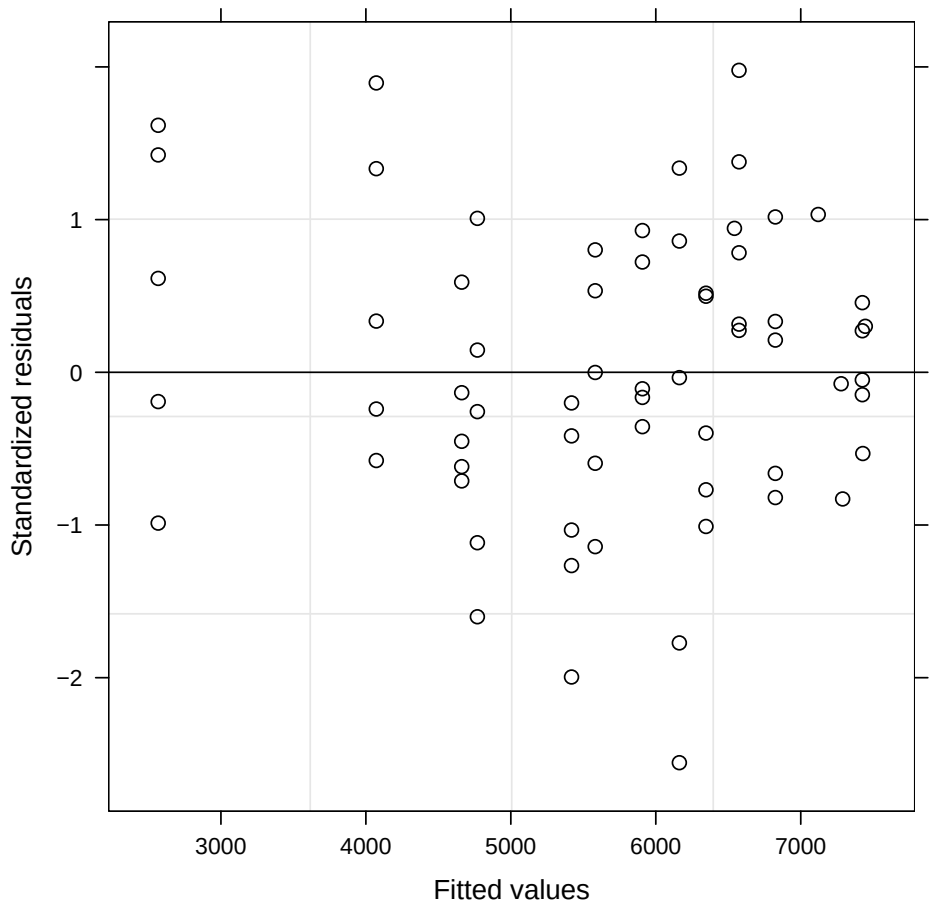
```
Error in gnls(cheact ~ tcmfn4(dose, tmpstds, lA = lA, lD = lD, lg = lg,  :
        Step halving factor reduced below minimum in NLS step
```

So, try setting `lg` to 0:

```
> Parm <- init1$start$beta
> Start <- Start0 <- c(getParms("^lA", Parm), Parm["lD"], Parm["lTr"])
> formals(tcmfn4)$tz <- -10
> formals(tcmfn4)$lg <- 0
> icnt <- 1
> Maxcnt <- 50
> repeat {
+     drmod1 <- try(gnls(cheact ~ tcmfn4(dose, tmpstds, lA = lA,
+         lD = lD, lTr = lTr), data = dta, params = list(lA ~ ftime -
+         1, lD ~ 1, lTr ~ 1), start = Start), silent = TRUE)
+     if (!inherits(drmod1, "try-error") || icnt > Maxcnt) {
+         if (icnt <= Maxcnt)
+             writeLines(paste("Successful in ", icnt, if (icnt >
+                 1)
+                 "tries"
+             else "try"))
+         else writeLines("Maxcnt exceeded")
+         break
+     }
+     icnt <- icnt + 1
+     Start <- Start0 * (1 + rnorm(length(Start0), sd = 0.2))
+ }
```

```
Successful in  3 tries
```

Residuals from the fit:



These look pretty uniform, but try adding `varPower()` to weights:

```
> Start <- Start0 <- coef(drmod1)
> formals(tcmfn4)$tz <- -10
> formals(tcmfn4)$lg <- 0
> icnt <- 1
> Maxcnt <- 50
> repeat {
+     drmod2 <- try(gnls(cheact ~ tcmfn4(dose, tmpstds, lA = lA,
+         lD = lD, lTr = lTr), data = dta, params = list(lA ~ ftime -
+         1, lD ~ 1, lTr ~ 1), weights = varPower(), start = Start),
+         silent = TRUE)
+     if (!inherits(drmod2, "try-error") || icnt > Maxcnt) {
+         if (icnt <= Maxcnt)
+             writeLines(paste("Successful in ", icnt, if (icnt >
+                 1)
+                 "tries"
+             else "try"))
+         else writeLines("Maxcnt exceeded")
+         break
+     }
+     icnt <- icnt + 1
```

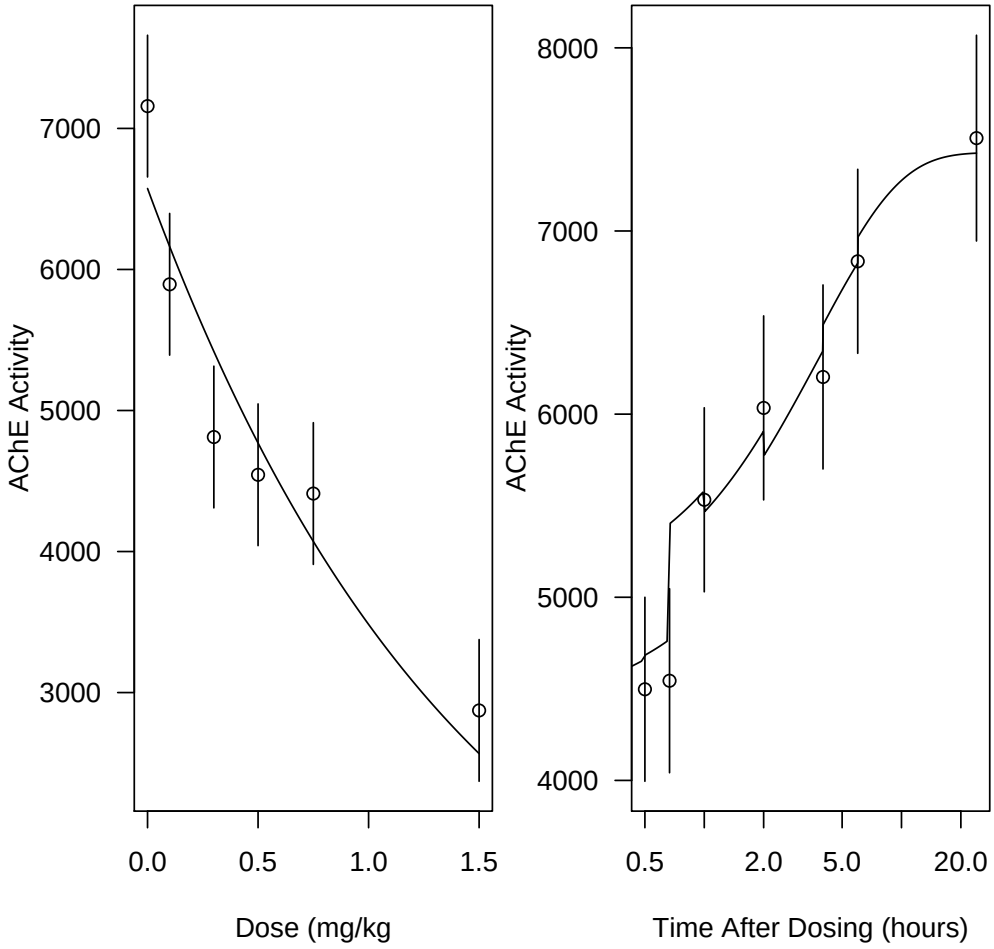
```
+     Start <- Start0 * (1 + rnorm(length(Start0), sd = 0.2))
+ }

Successful in  7 tries

> anova(drmod1, drmod2)

       Model df      AIC      BIC    logLik   Test  L.Ratio p-value
drmod1     1 10 1030.168 1051.912 -505.0842
drmod2     2 11 1101.720 1125.639 -539.8601 1 vs 2 69.55171  <.0001
```

It is not needed, so base further analyses on model fit in `drmod1`.

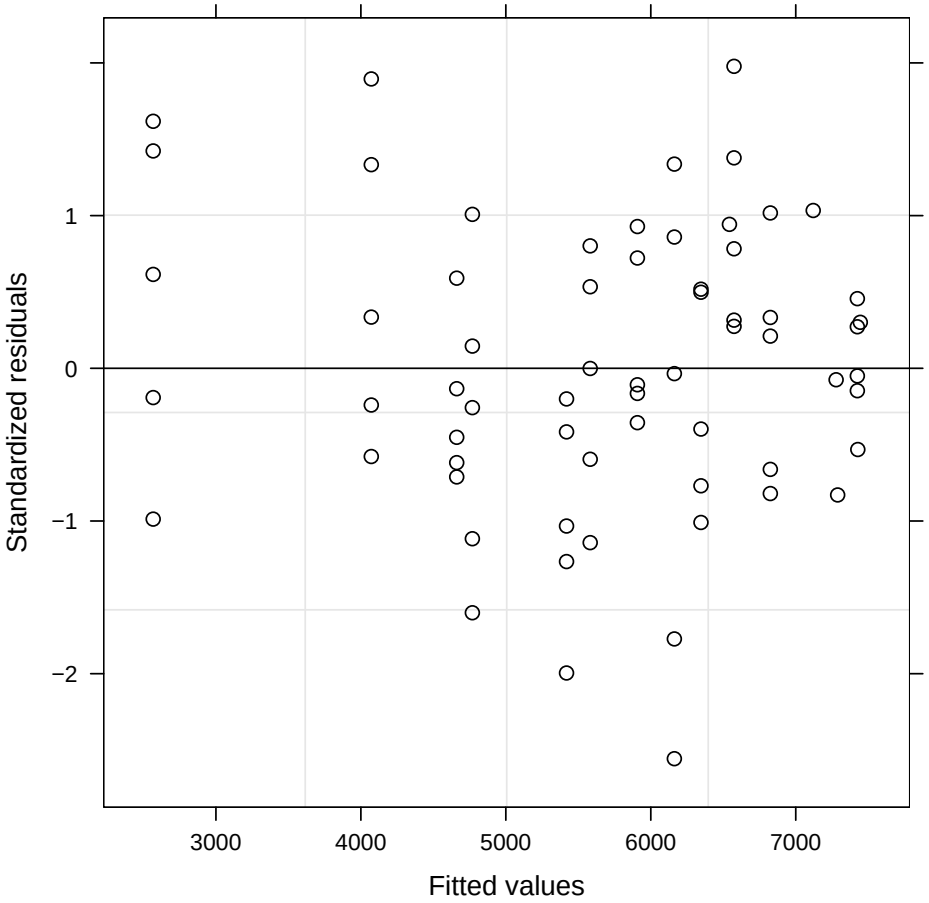
Can we simplify the control values?

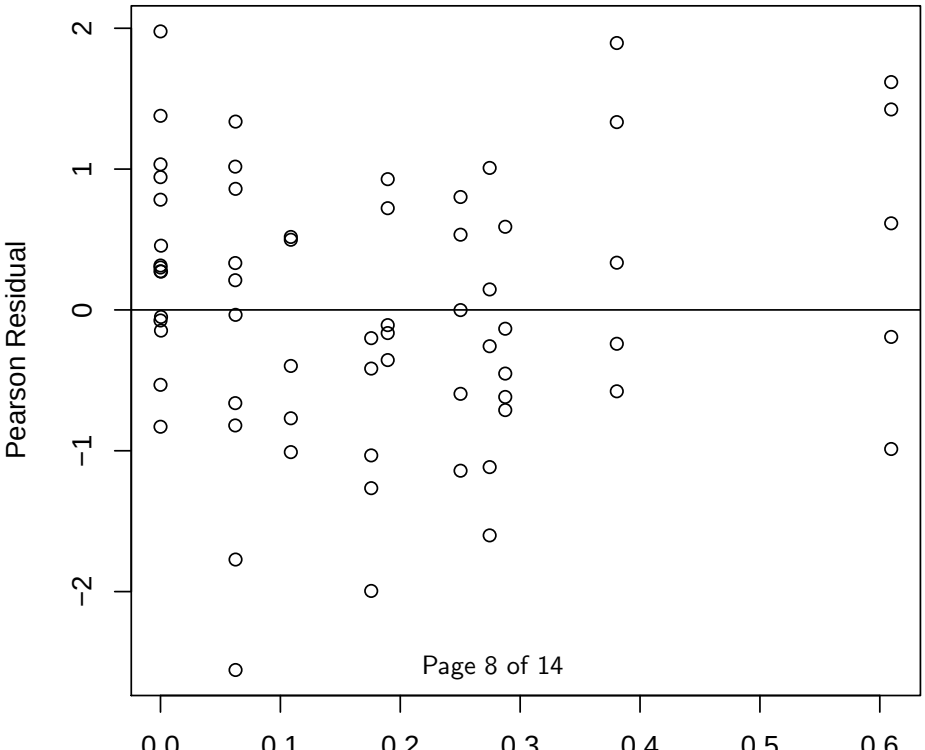
```
> nms <- names(getParms("^lA", coef(drmod1)))
> L <- diag(length(nms)) - matrix(1, nrow = length(nms), ncol = length(nms))/length(nms)
> colnames(L) <- nms
> L <- L[-nrow(L), ]
> anova(drmod1, L = L)

Denom. DF: 56
 F-test for linear combination(s)
  lA.ftime0.5 lA.ftime0.666666666666667  lA.ftime1  lA.ftime2  lA.ftime4
1   0.8571429                -0.1428571 -0.1428571 -0.1428571 -0.1428571
2  -0.1428571                 0.8571429 -0.1428571 -0.1428571 -0.1428571
3  -0.1428571                -0.1428571  0.8571429 -0.1428571 -0.1428571
4  -0.1428571                -0.1428571 -0.1428571  0.8571429 -0.1428571
5  -0.1428571                -0.1428571 -0.1428571 -0.1428571  0.8571429
6  -0.1428571                -0.1428571 -0.1428571 -0.1428571 -0.1428571
   lA.ftime6 lA.ftime24
1 -0.1428571 -0.1428571
2 -0.1428571 -0.1428571
3 -0.1428571 -0.1428571
4 -0.1428571 -0.1428571
5 -0.1428571 -0.1428571
6  0.8571429 -0.1428571
  numDF  F-value p-value
1     6 1.963859  0.0864
```

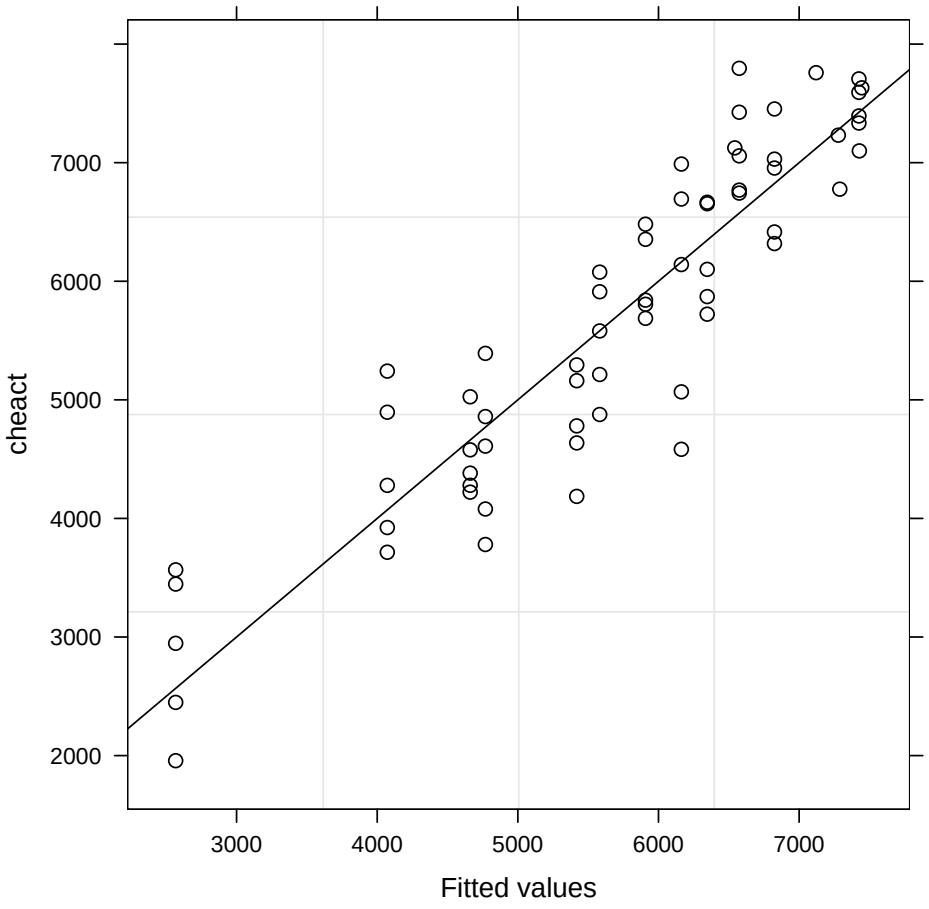
Not significant, but the P-value is small. Leave the model as it is. Diagnostic graphics for the model contained in `drmod1`: First, the dose-response and recovery curves with means and 95% confidence intervals from the data.

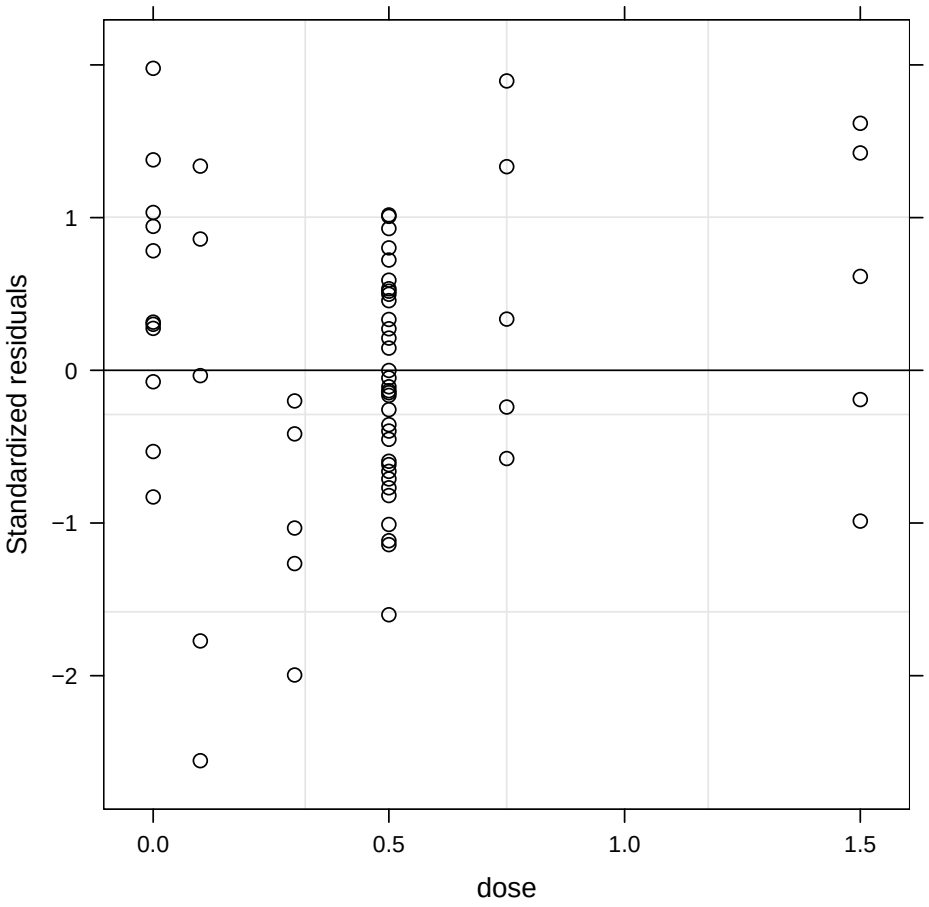


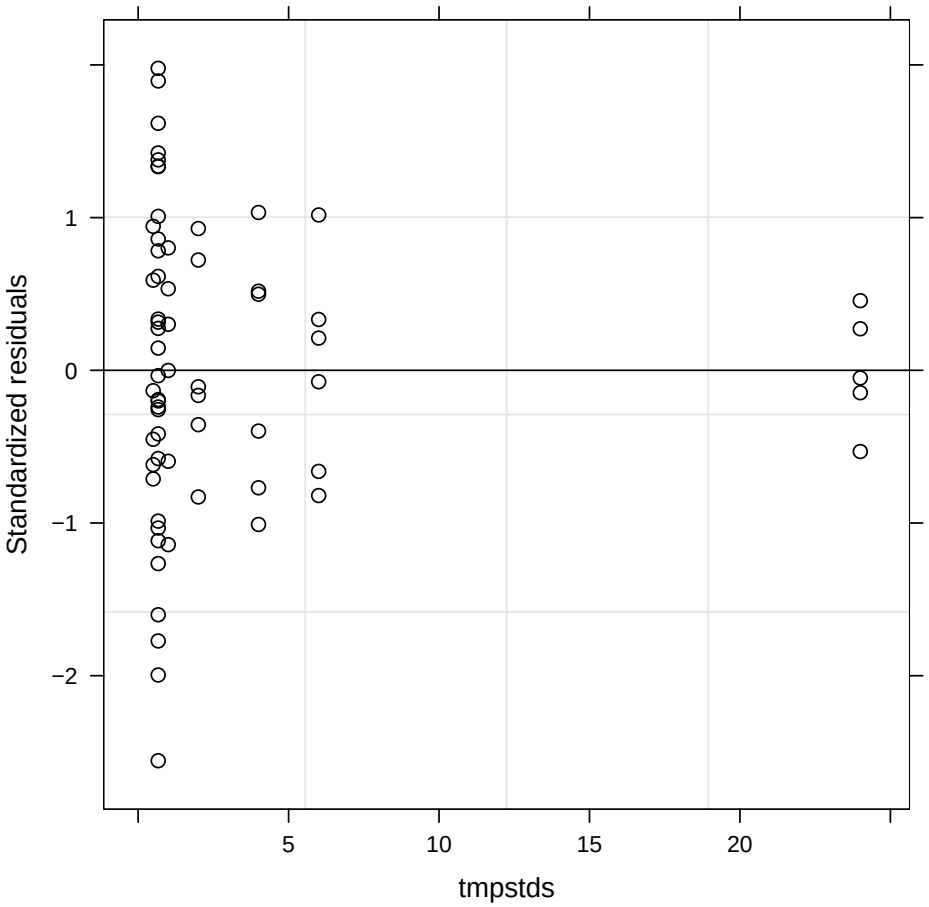
(The broken lines on the recovery curve reflect the variation of background among time points.

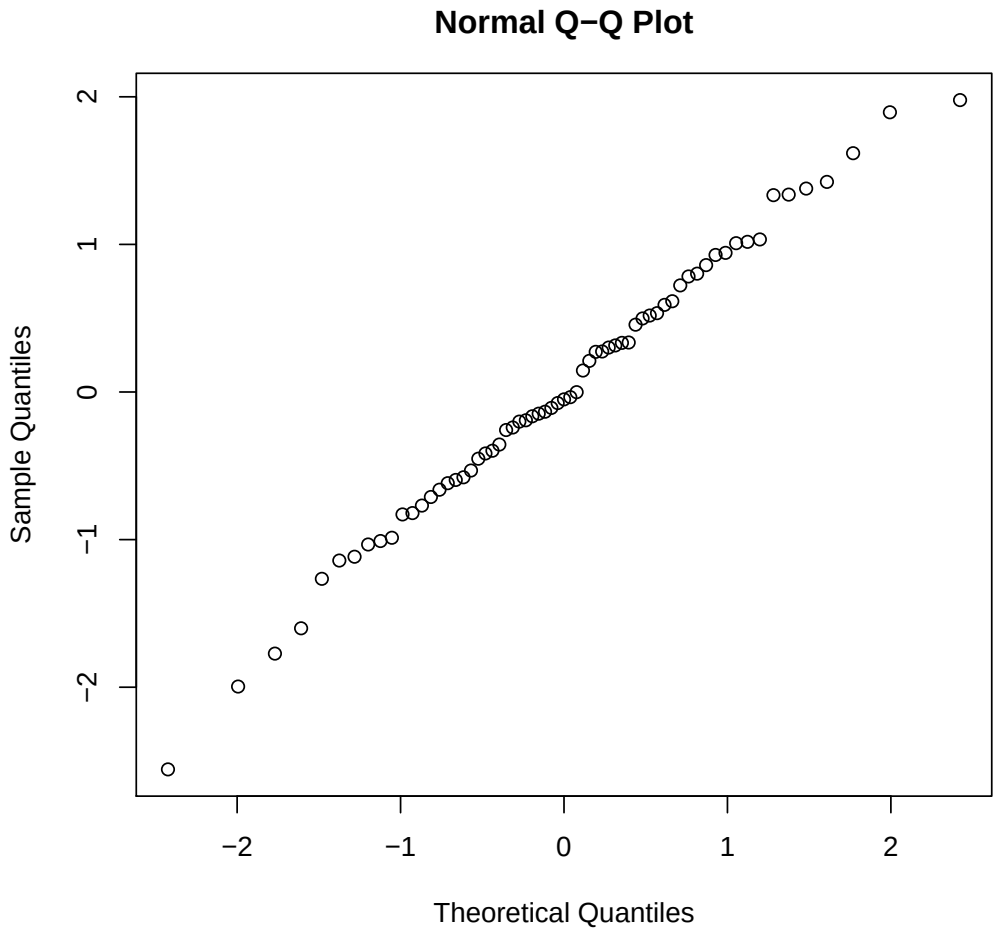

Standardized residuals
Fitted values
1
0
−1
−2
3000
4000
5000
6000
7000


Pearson Residual
2
1
0
−1
−2
0.0
0.1
0.2
0.3
0.4
0.5
0.6


cheact
Fitted values
2000
3000
4000
5000
6000
7000
3000
4000
5000
6000
7000


Standardized residuals
1
0
−1
−2
0.0
0.5
1.0
1.5
dose


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



Details of this final model fit:

```
> tTab <- summary(drmod1)$tTable
> Ints <- intervals(drmod1, which = "coef")$coef
> Ints90 <- intervals(drmod1, which = "coef", level = 0.9)$coef
> summary(drmod1)
```

```
Generalized nonlinear least squares fit
  Model: cheact ~ tcmfn4(dose, tmpstds, lA = lA, lD = lD, lTr = lTr)
  Data: dta
       AIC      BIC    logLik
  1030.168 1051.912 -505.0842

Coefficients:
                              Value Std.Error   t-value p-value
lA.ftime0.5                8.786087 0.0568036 154.67471    0.00
lA.ftime0.666666666666667  8.790889 0.0283404 310.18937    0.00
lA.ftime1                  8.915323 0.0471610 189.04033    0.00
lA.ftime2                  8.894136 0.0545739 162.97411    0.00
lA.ftime4                  8.870709 0.0626969 141.48572    0.00
lA.ftime6                  8.892673 0.0573162 155.15112    0.00
lA.ftime24                 8.913016 0.0372256 239.43265    0.00
lD                        -1.861980 0.1092862 -17.03765    0.00
```

```
lTr                        0.913899 0.5628195   1.62379    0.11

 Correlation:
                          lA.0.5 lA.0.6 lA.ft1 lA.ft2 lA.ft4 lA.ft6 lA.f24
lA.ftime0.666666666666667  0.280
lA.ftime1                  0.122  0.221
lA.ftime2                 -0.051  0.057  0.197
lA.ftime4                 -0.146 -0.050  0.167  0.503
lA.ftime6                 -0.162 -0.078  0.147  0.487  0.603
lA.ftime24                -0.008 -0.005  0.005  0.020  0.025  0.025
lD                        -0.468 -0.598 -0.261  0.110  0.311  0.347  0.017
lTr                       -0.250 -0.159  0.153  0.610  0.775  0.767  0.033
                          lD
lA.ftime0.666666666666667
lA.ftime1
lA.ftime2
lA.ftime4
lA.ftime6
lA.ftime24
lD
lTr                        0.533

Standardized residuals:
        Min          Q1         Med          Q3         Max
-2.55648299 -0.59556747 -0.04978245  0.61504517  1.97782822

Residual standard error: 617.7637
Degrees of freedom: 65 total; 56 residual
```

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` **(standard error)** -1.86198001807933 (0.109286209587773)

BMD (95% CI) 0.155364700699170 (0.124817001403045, 0.193388640585896)

BMDL, the one-sided lower 95% CL 0.129410773759597

`lTr (standard error)` 0.913899454653081 (0.562819502056976)

Recovery Half-life (95% CI) 2.49402894901173 (0.807700878216525, 7.70109401421386)

Save everything:

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

Save the results for incorporating into a database:

```
> carbofuran.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(carbofuran.oral.brain, file = file.path("..", "..", "01Summaries",
+     "carbofuran.oral.brain.RData"))
```