III. Appendices

B. Hazard/RPF

4. R Programs for the Revised Analysis

a. Part 1: Package RBMDS

RBMDS is a package of utility functions written to facilitate the analysis of the organophosphate acetylcholinesterase activity dose-response data. These functions were always available during the data analysis. See below in Part 2 for the .Rprofile file that guaranteed that.

```
### Generalization of exponential decreasing model
### A, B, and m are constrained to be strictly positive
CexpB <- function (Dose, A, B, m)
{
  ## exp(A)*(1/(1 + exp(-B)) + (exp(-B)/(1 + exp(-B)) * exp(-exp(m)*Dose)))
  .expr1 <- exp(A)
  .expr3 <- exp(-B)
  .expr4 <- 1 + .expr3
  .expr6 <- exp(m)
  .expr9 <- exp(-.expr6 * Dose)
  .expr10 <- .expr3 * .expr9
  .expr11 <- .expr10/.expr4
  .expr13 <- .expr1 * (1/.expr4 + .expr11)
  .expr14 <- .expr4^2
  .value <- .expr13
  .grad <- array(0, c(length(.value), 3), list(NULL, c("A",
                                                       "B", "m")))
  .grad[, "A"] <- .expr13
  .grad[, "B"] <- .expr1 * (.expr3/.expr14 - (.expr11 - .expr10 *
                                             .expr3/.expr14))
  .grad[, "m"] <- -.expr1 * (.expr3 * (.expr9 * (.expr6 *
                                               Dose))/.expr4)
  attr(.value, "gradient") <- .grad
  .value
}

### CexpB2: same as above, but gradient includes deriv wrt dose.

CexpB2 <- function (Dose, A, B, m)
{
  .expr1 <- exp(A)
  .expr3 <- exp(-B)
  .expr4 <- 1 + .expr3
  .expr6 <- .expr3/.expr4
  .expr7 <- exp(m)
  .expr10 <- exp(-.expr7 * Dose)
  .expr13 <- .expr1 * (1/.expr4 + .expr6 * .expr10)
  .expr18 <- .expr4^2
  .value <- .expr13
  .grad <- array(0, c(length(.value), 4), list(NULL, c("Dose",
                                                       "A", "B", "m")))
  .grad[, "Dose"] <- -.expr1 * (.expr6 * (.expr10 * .expr7))
  .grad[, "A"] <- .expr13
  .grad[, "B"] <- .expr1 * (.expr3/.expr18 - (.expr6 - .expr3 *
                                             .expr3/.expr18) * .expr10)
  .grad[, "m"] <- -.expr1 * (.expr6 * (.expr10 * (.expr7 *
                                                Dose)))
  attr(.value, "gradient") <- .grad
  .value
```

```
}


### CpkexpB: above model, with the additional assumption that there is
### saturable detoxification.  This is implemented by composing CexpB
### (now assuming that Dose in CexpB refers to internal dose), with
### a model that relates administered dose to internal dose:
###
### idose = 0.5*((Dose - S - D) + sqrt((Dose - S - D)^2 + 4*Dose*S))
###
### This model approaches a line with slope 1 and y-intercept -D as Dose
### increases to infinity.  The parameter 'S' controls the shape of the
### low-dose part of the curve.  For values close to 0, the curve looks
### very threshold-like; as S increases, the curve becomes more gradual.
### S and D must be positive
###
### The final function is (Dose refers to administered dose; exponentiation
### used to force positive parameters):
###
###
## exp(A)*(1/(1 + exp(-B)) + (exp(-B)/(1 + exp(-B)) * exp(-exp(m)*0.5*((Dose -
##  exp(S) - exp(D)) + sqrt((Dose - exp(S) - exp(D))^2 + 4*Dose*exp(S))))))
##

CpkexpB <- function (Dose, A, B, m, S, D)
{
  idose <- CpkB(Dose, S, D)
  .value <- CexpB2(idose, A, B, m)
  .grad <- array(0, c(length(.value), 5),
                 list(NULL,
                      c("A","B","m","S","D")))
  .grad[,c("A","B","m")] <- attr(.value,"gradient")[,c("A","B","m")]
  .grad[,c("S","D")] <- attr(.value,"gradient")[,"Dose"] *
    attr(idose, "gradient")[,c("S", "D")]
  attr(.value, "gradient") <- .grad
  .value
}

### CexpBS(dose, A, B, m, sex, fixed=NULL) allows fixed, a named list of
### vectors of fixed values for the model CexpB.  They can differ by sex.
### used, for example:
### nlme(model=chei ~ CexpBS(dose, A, m, sex,
###      fixed=list(B=c(F=-3.01, B=-2.78))), data=mydata, random=A+m~1 | mrid)


CexpBS <- function(dose, A, B, m, sex, fixed=NULL)
{
  callList <- vector("list",4)
  names(callList) <- c("Dose","A","B","m")
  sex <- switch(length(fixed) + 1,
                sex,
                m,
                B)

  callList[["Dose"]] <- dose
### Assume we will always estimate A
  callList[["A"]] <- A
  if ("B" %in% names(fixed)) {
    callList[["B"]] <- fixed[["B"]][as.character(sex)]
    callList[["m"]] <- if ("m" %in% names(fixed)) {
      fixed[["m"]][as.character(sex)]
    } else {
      B
    }
  } else {
    callList[["B"]] <- B
    callList[["m"]] <- if ("m" %in% names(fixed)) {
      fixed[["m"]][as.character(sex)]
    } else {
      m
    }
  }
  .value <- do.call("CexpB",callList)
  .grad <- attr(.value, "gradient")
  .grad <- .grad[,-match(names(fixed),colnames(.grad)),drop=FALSE]
  attr(.value,"gradient") <- .grad
  .value
}

### This is the same model, reparameterized so that, instead of 'm', we
### estimate 'BMD', for a BMR*100% reduction in mean response relative to
### control.
###
### Model is:
### ~ exp(A)*(1/(1 + exp(-B)) +
###           exp(-B)/(1 + exp(-B))*
###             exp(log(1 - BMR*(1 + exp(B))) * Dose*exp(-BMD)))

CexpBwD <- function (Dose, A, B, BMD, BMR=0.10)
{
  .expr1 <- exp(A)
  .expr3 <- exp(-B)
  .expr4 <- 1 + .expr3
  .expr6 <- .expr3/.expr4
  .expr7 <- exp(B)
  .expr10 <- 1 - BMR * (1 + .expr7)
  .expr14 <- exp(-BMD)
  .expr15 <- log(.expr10) * Dose * .expr14
  .expr16 <- exp(.expr15)
  .expr19 <- .expr1 * (1/.expr4 + .expr6 * .expr16)
  .expr20 <- .expr4^2
  .value <- .expr19
  .grad <- array(0, c(length(.value), 3), list(NULL, c("A",
                                                       "B", "BMD")))
  .grad[, "A"] <- .expr19
  .grad[, "B"] <- .expr1 * (.expr3/.expr20 - (.expr6 * (.expr16 *
                                                        (BMR * .expr7/.expr10 * Dose *
.expr14)) + (.expr6 - .expr3 * .expr3/.expr20) * .expr16))
  .grad[, "BMD"] <- -.expr1 * (.expr6 * (.expr16 * .expr15))
  attr(.value, "gradient") <- .grad
  .value
}

### Include the derivative wrt Dose, to combine with the Pk model:

CexpBwD2 <- function (Dose, A, B, BMD, BMR=0.10)
{
  .expr1 <- exp(A)
  .expr3 <- exp(-B)
  .expr4 <- 1 + .expr3
  .expr6 <- .expr3/.expr4
  .expr7 <- exp(B)
  .expr10 <- 1 - BMR * (1 + .expr7)
  .expr11 <- log(.expr10)
  .expr14 <- exp(-BMD)
  .expr15 <- .expr11 * Dose * .expr14
  .expr16 <- exp(.expr15)
  .expr19 <- .expr1 * (1/.expr4 + .expr6 * .expr16)
  .expr24 <- .expr4^2
  .value <- .expr19
  .grad <- array(0, c(length(.value), 4), list(NULL, c("Dose",
                                                       "A", "B", "BMD")))
  .grad[, "Dose"] <- .expr1 * (.expr6 * (.expr16 * (.expr11 *
                                                    .expr14)))
  .grad[, "A"] <- .expr19
  .grad[, "B"] <- .expr1 * (.expr3/.expr24 - (.expr6 * (.expr16 *
                                                      (BMR * .expr7/.expr10 * Dose *
.expr14)) + (.expr6 - .expr3 * .expr3/.expr24) * .expr16))
  .grad[, "BMD"] <- -.expr1 * (.expr6 * (.expr16 * .expr15))
  attr(.value, "gradient") <- .grad
  .value
}


### CexpBwDS(dose, A, B, BMD, sex, fixed=NULL) allows fixed, a named list of
### vectors of fixed values for the model CexpBwD.  They can differ by sex.
### used, for example:
### nlme(model=chei ~ CexpBwDS(dose, A, BMD, sex,
###      fixed=list(B=c(F=-3.01, B=-2.78))), data=mydata, random=A+m~1 | mrid)
### This implementation assumes B is the only fixed parameter.

CexpBwDS <- function(dose, A, BMD, sex, fixed=NULL, BMR=0.10)
{
  callList <- vector("list",5)
  names(callList) <- c("Dose","A","B","BMD","BMR")
  callList[["Dose"]] <- dose
  callList[["A"]] <- A
  callList[["B"]] <- fixed[["B"]][as.character(sex)]
  callList[["BMD"]] <- BMD
  callList[["BMR"]] <- BMR
  do.call("CexpBwD",callList)
}

### CpkexpBwD: pk combined with the exp model in terms of BMD:

CpkexpBwD <- function (Dose, A, B, BMD, S, D, BMR=0.10)
{
  idose <- CpkB(Dose, S, D)
  .value <- CexpBwD2(idose, A, B, BMD, BMR=BMR)
  .grad <- array(0, c(length(.value), 5),
                 list(NULL,
                      c("A","B","BMD","S","D")))
  .grad[,c("A","B","BMD")] <- attr(.value,"gradient")[,c("A","B","BMD")]
  .grad[,c("S","D")] <- attr(.value,"gradient")[,"Dose"] *
    attr(idose, "gradient")[,c("S", "D")]
  attr(.value, "gradient") <- .grad
  .value
}

### CpkexpBS: pk combined with exp model, with fixed B and/or S values:

CpkexpBS <- function(dose, A, B, m, S, D, sex, fixed=NULL)
{
  callList <- vector("list",6)
  names(callList) <- c("Dose","A","B","m","S","D")
  callList[["Dose"]] <- dose
  callList[["A"]] <- A
  if (length(fixed) == 0) sx <- sex
  if (length(fixed) == 1) sx <- D
  if (length(fixed) == 2) sx <- S
  if (length(fixed) == 3) sx <- m
  if ("B" %in% names(fixed)) {
    callList[["B"]] <- fixed[["B"]][as.character(sx)]
    callList[["m"]] <- B
    if ("S" %in% names(fixed)) {
      callList[["S"]] <- fixed[["S"]][as.character(sx)]
      if ("D" %in% names(fixed)) {
        callList[["D"]] <- fixed[["D"]][as.character(sx)]
      } else {
        callList[["D"]] <- m
      }
    } else {
      callList[["S"]] <- m
      if ("D" %in% names(fixed)) {
        callList[["D"]] <- fixed[["D"]][as.character(sx)]
      } else {
        callList[["D"]] <- S
      }
    }
  } else {
    callList[["B"]] <- B
    callList[["m"]] <- m
    if ("S" %in% names(fixed)) {
      callList[["S"]] <- fixed[["S"]][as.character(sx)]
      if ("D" %in% names(fixed)) {
        callList[["D"]] <- fixed[["D"]][as.character(sx)]
      } else {
        callList[["D"]] <- S
      }
    } else {
      callList[["S"]] <- S
      if ("D" %in% names(fixed)) {
        callList[["D"]] <- fixed[["D"]][as.character(sx)]
      } else {
        callList[["D"]] <- D
      }
    }
  }
  do.call("CpkexpB",callList)
}

### Compute log(BMD) (for a BMR * 100% decrease in the mean) for the
### exponential
### model, based on the parameter transformations used here.
### This returns the gradient of log(BMD), to use in computing standard
### errors.
###
### log(BMD) <- expression(log(-log(1 - BMR*(1 + exp(B)))) - m)
### in terms of the transformation used in the CexpB models.
### fixed is a vector of strings, listing the parameters that should not
### be in the gradient.

CexplBMD <- function (BMR, m, B, fixed=NULL)
{
  .expr1 <- exp(B)
  .expr4 <- 1 - BMR * (1 + .expr1)
  .expr5 <- log(.expr4)
  .value <- log(-.expr5) - m
  .grad <- array(0, c(length(.value), 2), list(NULL, c("m",
                                                       "B")))
  .grad[, "m"] <- -1
  .grad[, "B"] <- -BMR * .expr1/.expr4/.expr5
  if (!is.null(fixed) > 0) {
    .grad <- .grad[,-match(fixed,colnames(.grad)),drop=FALSE]
  }
  attr(.value, "gradient") <- .grad
  .value
}

### Essentially the same function, but takes as input our model object
### (xx) and returns a list with elements lBMD and lBMD.se,
### each with components for "F" and "M".


explBMD <- function(object, BMR)
{
  fit <- object$Fitm
  if (inherits(fit, "nlme")) {
    Sigma <- fit$varFix
    Coefs <- fit$coefficients$fixed
  } else {
    Sigma <- fit$varBeta
    Coefs <- fit$coefficients
  }
### Fixed gives the fixed parameters
  tmp <- names(fit$call$model[[3]])
  Fixed <- NULL
  if ("fixed" %in% tmp) {
    tmp <- names(fit$call$model[[3]][["fixed"]])
    Fixed <- c(Fixed,c("m","B")[c("m","B") %in% tmp])
  }
  m.F <- if ("m" %in% Fixed)
    fit$call$model[[3]][["fixed"]][["m"]][["F"]]
  else
    if ("m.sexF" %in% names(Coefs)) Coefs["m.sexF"] else Coefs["m"]
  m.M <- if ("m" %in% Fixed)
    fit$call$model[[3]][["fixed"]][["m"]][["M"]]
  else
    if ("m.sexM" %in% names(Coefs)) Coefs["m.sexM"] else Coefs["m"]

  B.F <- if ("B" %in% Fixed)
    fit$call$model[[3]][["fixed"]][["B"]][["F"]]
  else
    if ("B.sexF" %in% names(Coefs)) Coefs["B.sexF"] else Coefs["B"]
  B.M <- if ("B" %in% Fixed)
    fit$call$model[[3]][["fixed"]][["B"]][["M"]]
  else
    if ("B.sexM" %in% names(Coefs)) Coefs["B.sexM"] else Coefs["B"]
### Females
  lBMD.F <- CexplBMD(BMR=BMR,m=m.F,B=B.F,fixed=Fixed) + log(object$Dosescale)
### Males
  lBMD.M <- CexplBMD(BMR=BMR,m=m.M,B=B.M,fixed=Fixed) + log(object$Dosescale)

  ## Get the standard errors
  ## 1) Get the 'Female' and 'Male' Sigmas
  indx <- match(c("m.sexF","B.sexF"),rownames(Sigma),nomatch=0)
  Sigma.F <- Sigma[indx,indx,drop=FALSE]

  ## Strip off the '.F' from the rownames and colnames
  rownames(Sigma.F) <- colnames(Sigma.F) <- sub("\\.sexF","",rownames(Sigma.F))

  indx <- match(c("m.sexM","B.sexM"),rownames(Sigma),nomatch=0)
  Sigma.M <- Sigma[indx,indx,drop=FALSE]

  ## Strip off the '.M' from the rownames and colnames
  rownames(Sigma.M) <- colnames(Sigma.M) <- sub("\\.sexM","",rownames(Sigma.M))

  ## Get the 'Female' and 'Male' gradients

  grad.F <- attr(lBMD.F,"gradient")
  grad.F <- grad.F[,rownames(Sigma.F)]
  grad.M <- attr(lBMD.M,"gradient")
  grad.M <- grad.M[,rownames(Sigma.M)]
  list(lBMD = c(F=c(lBMD.F),M=c(lBMD.M)),
       lBMD.se = c(F=sqrt(t(grad.F) %*% Sigma.F %*% grad.F),
         M=sqrt(t(grad.M) %*% Sigma.M %*% grad.M)))
}

### Compute BMD (for a BMR * 100% decrease in the mean) for the
### exponential
### model, based on the parameter transformations used here.
### This returns the gradient of BMD, to use in computing standard
### errors.
###
### BMD <- expression(-log(1 - BMR*(1 + exp(B)))*exp( - m ))
### in terms of the transformation used in the CexpB models.
### fixed is a vector of strings, listing the parameters that should not
### be in the gradient.

CexpBMD <- function (BMR, m, B, fixed)
```

```
{
  .expr1 <- exp(B)
  .expr4 <- 1 - BMR * (1 + .expr1)
  .expr5 <- log(.expr4)
  .expr8 <- exp(-m)
  .value <- -.expr5 * .expr8
  .grad <- array(0, c(length(.value), 2), list(NULL, c("m",
                                                       "B")))
  .grad[, "m"] <- .expr5 * .expr8
  .grad[, "B"] <- BMR * .expr1/.expr4 * .expr8
  if (!is.null(fixed) > 0) {
    .grad <- .grad[,-match(fixed,colnames(.grad)),drop=FALSE]
  }
  attr(.value, "gradient") <- .grad
  .value
}

### Compute lBMD and its standard error for pkexpBS
### This calculates the value for both sexes at the same time

pkexpSlBMD.se <- function(object,BMR)
{
  fitpk <- object$Fitpk
  if (inherits(fitpk, "nlme")) {
    Sigma <- fitpk$varFix
    Coefs <- fitpk$coefficients$fixed
  } else {
    Sigma <- fitpk$varBeta
    Coefs <- fitpk$coefficients
  }

### Fixed1 gives the fixed parameters for the CexpBMD part (i.e., 'B')
### Fixed2 gives the fixed parameters for the CpkBMD part (i.e., 'S')

  tmp <- names(fitpk$call$model[[3]])
  Fixed1 <- Fixed2 <- NULL
  if ("fixed" %in% tmp) {
    tmp <- names(fitpk$call$model[[3]][["fixed"]])
    Fixed1 <- c(Fixed1,c("m","B")[c("m","B") %in% tmp])
    Fixed2 <- c(Fixed2,c("S","D")[c("S","D") %in% tmp])
  }

  S.F <- if ("S" %in% Fixed2)
    fitpk$call$model[[3]][["fixed"]][["S"]][["F"]]
  else
    if ("S.sexF" %in% names(Coefs)) Coefs["S.sexF"] else Coefs["S"]
  S.M <- if ("S" %in% Fixed2)
    fitpk$call$model[[3]][["fixed"]][["S"]][["M"]]
  else
    if ("S.sexM" %in% names(Coefs)) Coefs["S.sexM"] else Coefs["S"]

  D.F <- if ("D" %in% Fixed2)
    fitpk$call$model[[3]][["fixed"]][["D"]][["F"]]
  else
    if ("D.sexF" %in% names(Coefs)) Coefs["D.sexF"] else Coefs["D"]
  D.M <- if ("D" %in% Fixed2)
    fitpk$call$model[[3]][["fixed"]][["D"]][["M"]]
  else
    if ("D.sexM" %in% names(Coefs)) Coefs["D.sexM"] else Coefs["D"]

  m.F <- if ("m" %in% Fixed1)
    fitpk$call$model[[3]][["fixed"]][["m"]][["F"]]
  else
    if ("m.sexF" %in% names(Coefs)) Coefs["m.sexF"] else Coefs["m"]
  m.M <- if ("m" %in% Fixed1)
    fitpk$call$model[[3]][["fixed"]][["m"]][["M"]]
  else
    if ("m.sexM" %in% names(Coefs)) Coefs["m.sexM"] else Coefs["m"]
  B.F <- if ("B" %in% Fixed1)
    fitpk$call$model[[3]][["fixed"]][["B"]][["F"]]
  else
    if ("B.sexF" %in% names(Coefs)) Coefs["B.sexF"] else Coefs["B"]
  B.M <- if ("B" %in% Fixed1)
    fitpk$call$model[[3]][["fixed"]][["B"]][["M"]]
  else
    if ("B.sexM" %in% names(Coefs)) Coefs["B.sexM"] else Coefs["B"]

### Females
  idose.F <- CexpBMD(BMR,m.F,B.F,fixed=Fixed1)
  lBMD.F <- CpkBi(as.vector(idose.F),S.F,D.F,fixed=Fixed2) +
    log(object$Dosescale)

### Males
  idose.M <- CexpBMD(BMR,m.M,B.M,fixed=Fixed1)
  lBMD.M <- CpkBi(as.vector(idose.M),S.M,D.M,fixed=Fixed2) +
    log(object$Dosescale)

  ## Get the standard errors
  ## 1) Get the 'Female' and 'Male' Sigmas
  indx <- match(c("m.sexF","B.sexF","S","D"),rownames(Sigma),nomatch=0)
  Sigma.F <- Sigma[indx,indx,drop=FALSE]

  ## Strip off the '.F' from the rownames and colnames
  rownames(Sigma.F) <- colnames(Sigma.F) <- sub("\\.sexF","",rownames(Sigma.F))

  indx <- match(c("m.sexM","B.sexM","S","D"),rownames(Sigma),nomatch=0)
  Sigma.M <- Sigma[indx,indx,drop=FALSE]

  ## Strip off the '.M' from the rownames and colnames
  rownames(Sigma.M) <- colnames(Sigma.M) <- sub("\\.sexM","",rownames(Sigma.M))

  ## 2) Get the 'Female' and 'Male' gradients

  grad.F <- c(attr(idose.F,"gradient")*attr(lBMD.F,"gradient")[,"idose"],
              attr(lBMD.F,"gradient")[,-match("idose",
                                          colnames(attr(lBMD.F,
                                                        "gradient"))),
                                    drop=FALSE])
  nm <- c(colnames(attr(idose.F,"gradient")),
          colnames(attr(lBMD.F,"gradient")))
  names(grad.F) <- nm[-match("idose",nm)]

  ## Make sure the elements are in the right order for Sigma.F

  grad.F <- grad.F[rownames(Sigma.F)]

  grad.M <- c(attr(idose.M,"gradient")*attr(lBMD.M,"gradient")[,"idose"],
              attr(lBMD.M,"gradient")[,-match("idose",
                                          colnames(attr(lBMD.M,
                                                        "gradient"))),
                                    drop=FALSE])
  nm <- c(colnames(attr(idose.M,"gradient")),
          colnames(attr(lBMD.M,"gradient")))
  names(grad.M) <- nm[-match("idose",nm)]

  ##Make sure the elements are in the right order for Sigma.M

  grad.M <- grad.M[rownames(Sigma.M)]

  list(lBMD=c(F=as.vector(lBMD.F),M=as.vector(lBMD.M)),
       lBMD.se=c(F = sqrt(t(grad.F) %*% Sigma.F %*% grad.F),
         M = sqrt(t(grad.M) %*% Sigma.M %*% grad.M)))
}

### CpkB: a low-dose modification to simulate the effect of first-pass
### metabolism on the relationship between administered dose and
### internal dose.
### This maps administered dose (D) to internal dose (CpkB)

CpkB <- function (dose, S, D)
{
  .expr1 <- exp(S)
  .expr3 <- exp(D)
  .expr4 <- dose - .expr1 - .expr3
  .expr7 <- 4 * dose * .expr1
  .expr8 <- .expr4^2 + .expr7
  .expr15 <- .expr8^-0.5
  .value <- 0.5 * (.expr4 + sqrt(.expr8))
  .grad <- array(0, c(length(.value), 2), list(NULL, c("S",
                                                       "D")))
  .grad[, "S"] <- 0.5 * (0.5 * ((.expr7 - 2 * (.expr1 * .expr4)) *
                                .expr15) - .expr1)
  .grad[, "D"] <- -0.5 * (0.5 * (2 * (.expr3 * .expr4) * .expr15) +
                          .expr3)
  attr(.value, "gradient") <- .grad
  .value
}

### and this is the inverse function: maps internal dose to the corresponding
### administered dose.
### CpkBi <- expression((idose^2 + idose*(exp(S) + exp(D)))/(idose + exp(S)))
###
### Since I want to give log(BMD), use:
### CpkBi <- expression(log(idose^2 + idose*(exp(S) + exp(D))) - log(idose + ###
   exp(S)))
### fixed: a vector of strings of parameters that will not be included in
### the gradient

CpkBi <- function (idose, S, D, fixed=NULL)
{
  .expr2 <- exp(S)
  .expr3 <- exp(D)
  .expr4 <- .expr2 + .expr3
  .expr6 <- idose^2 + idose * .expr4
  .expr8 <- idose + .expr2
  .value <- log(.expr6) - log(.expr8)
  .grad <- array(0, c(length(.value), 3), list(NULL, c("idose",
                                                       "S", "D")))
  .grad[, "idose"] <- (2 * idose + .expr4)/.expr6 - 1/.expr8
  .grad[, "S"] <- idose * .expr2/.expr6 - .expr2/.expr8
  .grad[, "D"] <- idose * .expr3/.expr6
  if (!is.null(fixed))
    .grad <- .grad[,-match(fixed,colnames(.grad)),drop=FALSE]
  attr(.value, "gradient") <- .grad
  .value
}

mypkPredict <- function (object, newdata, level)
{
  fparms <- object$coefficients$fixed
  rparms <- object$coefficients$random
  B.est <- !("fixed" %in% names(object$call$model[[3]])) ||
    !("B" %in% names(object$call$model[[3]][["fixed"]]))
  D.est <- !("fixed" %in% names(object$call$model[[3]])) ||
    !("D" %in% names(object$call$model[[3]][["fixed"]]))

  indx <- paste("A.", "s.U", as.character(newdata$s.U), sep = "")
  A <- fparms[indx]
  if (B.est) {
    indx <- paste("B.", "sex", as.character(newdata$sex),
                  sep = "")
    B <- fparms[indx]
  }
  else {
    B <- object$call$model[[3]]["fixed"][[1]]$B[as.character(newdata$sex)]
  }
  indx <- paste("m.", "sex", as.character(newdata$sex),
                sep = "")
  m <- fparms[indx]
### S is always fixed in my application
  S <- eval(object$call$model[[3]][["fixed"]][["S"]])[as.character(newdata$sex)]

### for simplicity, I'm assuming what we actually did, that is, model D ~ 1
### but fixed=list(D=c(F=,M=))

  if (D.est) {
    D <- fparms["D"]
  } else {
    D <-       eval(object$call$model[[3]][["fixed"]][["D"]])[
                                  as.character(newdata$sex)]
  }
  if (level >= 1) {
    grpname <- names(rparms)[1]
    if ("A.(Intercept)" %in% colnames(rparms[[1]]))
      A <- A + rparms[[1]][as.character(newdata[,grpname]),
                            "A.(Intercept)"]
    if (B.est)
      B <- B + rparms[[1]][as.character(newdata[,grpname]),
                            "B.(Intercept)"]
    m <- m + rparms[[1]][as.character(newdata[,grpname]),  "m.(Intercept)"]
  }
  if (level == 2) {
    indx <- paste(as.character(newdata$mrid), as.character(newdata$set),
                  sep = "/")
    if ("A.(Intercept)" %in% colnames(rparms[[2]]))
      A <- A + rparms[[2]][indx, "A.(Intercept)"]
    if (B.est)
      B <- B + rparms[[2]][indx, "B.(Intercept)"]
    m <- m + rparms[[2]][indx, "m.(Intercept)"]
  }
  if (B.est) {
    if (D.est) {
      eval(substitute(CpkexpBS(newdata$Dose.scaled, A, B, m, D,
                               newdata$sex, fixed=xxxx),
                      list(xxxx=eval(object$call$model[[3]][["fixed"]]))))
    }
    else {
      eval(substitute(CpkexpBS(newdata$Dose.scaled, A, B, m,
                               newdata$sex, fixed=xxxx),
                      list(xxxx=eval(object$call$model[[3]][["fixed"]]))))

    }
  } else {
    if (D.est) {
      eval(substitute(CpkexpBS(newdata$Dose.scaled, A, m, D, yyyy,
                               fixed=xxxx),
                      list(xxxx=eval(object$call$model[[3]][["fixed"]]),
                           yyyy=newdata$sex)))

    }
    else {
      eval(substitute(CpkexpBS(newdata$Dose.scaled, A, m,
                               newdata$sex, fixed=xxxx),
                      list(xxxx=eval(object$call$model[[3]][["fixed"]]))))

    }
  }
}
myPredict <- function(object, newdata,level)
{
  fparms <- object$coefficients$fixed
  rparms <- object$coefficients$random
  ## Did we estimate Bs?
  B.est <- !("fixed" %in% names(object$call$model[[3]]))
  ## is  this 'm' or 'BMD'?
  is.m <- "m.sexF" %in% names(fparms)

  indx <- paste("A.","s.U",as.character(newdata$s.U),sep="")
  A <- fparms[indx]
  if (B.est) {
    indx <- paste("B.","sex",as.character(newdata$sex),sep="")
    B <- fparms[indx]
  } else {
    B <- object$call$model[[3]]["fixed"][[1]]$B[as.character(newdata$sex)]
  }
  if (is.m) {
    indx <- paste("m.","sex",as.character(newdata$sex),sep="")
    thirdcol <- "m.(Intercept)"
  } else {
    indx <- paste("BMD.sex",as.character(newdata$sex),sep="")
    thirdcol <- "BMD.(Intercept)"
  }
  m <- fparms[indx]
  if (level >= 1) {
    grpname <- names(rparms)[1]
    if ("A.(Intercept)" %in% colnames(rparms[[1]]))
      A <- A + rparms[[1]][as.character(newdata[,grpname]),"A.(Intercept)"]
    if (B.est)
      B <- B + rparms[[1]][as.character(newdata[,grpname]),"B.(Intercept)"]
    m <- m + rparms[[1]][as.character(newdata[,grpname]),thirdcol]
  }
  if (level == 2) {
    indx <- paste(as.character(newdata$mrid),as.character(newdata$set),sep="/")
    if ("A.(Intercept)" %in% colnames(rparms[[2]]))
      A <- A + rparms[[2]][indx,"A.(Intercept)"]
    if (B.est)
      B <- B + rparms[[2]][indx,"B.(Intercept)"]
    m <- m + rparms[[2]][indx,thirdcol]
  }
  if (B.est) {
    if (is.m) {
      CexpB(newdata$Dose.scaled,A,B,m)
    } else {
      CexpBwD(newdata$Dose.scaled,A,B,m)
    }
  } else {
    if (is.m) {
      eval(substitute(CexpBS(newdata$Dose.scaled,A,m,yyyy,fixed=xxxx),
                      list(xxxx=object$call$model[[3]]["fixed"][[1]],
                           yyyy=newdata$sex)))
    } else {
      eval(substitute(CexpBwDS(newdata$Dose.scaled,A,m,yyyy,fixed=xxxx),
                      list(xxxx=object$call$model[[3]]["fixed"][[1]],
                           yyyy=newdata$sex)))
    }
  }

}
### Function to produce a "phony" dataset for input into gls and gnls
### Dose, N, M, and SD can be vectors of the same length
### DoseName and RespName are strings that give the names for the
### corresponding variables

PhonyDF <- function(Dose,N,M,SD,DoseName,RespName,chem=NULL,ChemName=NULL)
{
  tmp <- data.frame(rep(Dose,N),
                    qlnorm01(N,M,SD))
  names(tmp) <- c(DoseName,RespName)
  if (!is.null(ChemName)) tmp[,ChemName] <- factor(rep(chem,N))
  tmp
```

```
}

### from Bill Venables, R-list

plotCI <- function (x, y = NULL, uiw, liw = uiw, aui=NULL, ali=aui,
                    err="y", ylim=NULL, xlim=NULL, sfrac = 0.01, gap=0,
                    add=FALSE,
                    col=par("col"), lwd=par("lwd"), slty=par("lty"),
                    xlab=NULL,
                    ylab=NULL, main="", pt.bg = NA, scol=col,
                    axes=TRUE, ...)
{
  if (is.list(x)) {
    y <- x$y
    x <- x$x
  }
  if (is.null(y)) {
    if (is.null(x))
      stop("both x and y NULL")
    y <- as.numeric(x)
    x <- seq(along = x)
  }
  if (missing(xlab)) xlab <- deparse(substitute(x))
  if (missing(ylab)) ylab <- deparse(substitute(y))
  if (missing(uiw)) {                   ## absolute limits
    ui <- aui
    li <- ali
  }
  else {                                ## relative limits
    if (err=="y") z <- y else z <- x
    ui <- z + uiw
    li <- z - liw
  }
  if (err=="y" & is.null(ylim))
    ylim <- range(c(y, ui, li), na.rm=TRUE)
  else
    if (err=="x" & is.null(xlim))
      xlim <- range(c(x, ui, li), na.rm=TRUE)
  if (!add)
    plot(x, y, ylim = ylim, xlim = xlim, col=col, lwd=lwd,
         xlab=xlab, ylab=ylab,
         main=main, type="n", axes=axes, ...)
  if (gap==TRUE) gap <- 0.01            ## default gap size
                                        #  ul <- c(li, ui)
  if (err=="y") {
    gap <- rep(gap,length(x))*diff(par("usr")[3:4])
    arrows(x , li, x, pmax(y-gap,li), angle=90,length=sfrac*par("pin")[1],
           col=scol, lwd=lwd, lty=slty,code=1)
    arrows(x , ui, x, pmin(y+gap,ui), angle=90,length=sfrac*par("pin")[1],
           col=scol, lwd=lwd, lty=slty,code=1)
  }
  else if (err=="x") {
    gap <- rep(gap,length(x))*diff(par("usr")[1:2])
    arrows(li, y, pmax(x-gap,li), y, angle=90,length=sfrac*par("pin")[2],
           col=scol, lwd=lwd, lty=slty, code=1)
    arrows(ui, y, pmin(x+gap,ui), y, angle=90,length=sfrac*par("pin")[2],
           col=scol, lwd=lwd, lty=slty, code=1)
  }
  ## _now_ draw the points (in case we want to have "bg" set for points)
  points(x, y, col=col, lwd=lwd, bg=pt.bg, ...)
  invisible(list(x = x, y = y))
}
### Function to generate N approximately lognormal quantiles with exactly
### mean M and sd SD.  Does reasonable things if N,M,and SD are vectors
### that have the same length, and exits otherwise.  If N[i] == 1,
### then just returns the value of M[i].  If SD[i] == 0, returns
### rep(M[i],N[i]).
```

```
qlnorm01 <- function(N,M,SD)
{
  if((length(N) != length(M) || (length(N) != length(SD))))
    stop("N, M, and SD must have the same length")
  out <- NULL
  for (i in 1:length(N)) {
    if (N[i] > 1) {
      if (SD[i] > 0) {
        sdlog <- sqrt(log(SD[i]^2/M[i]^2 + 1))
        mnlog <- log(M[i]) - sdlog^2/2
        y <- qlnorm(ppoints(N[i]),mnlog,sdlog)
        sd <- sqrt(var(y))
        mn <- mean(y)
        out <- c(out,SD[i]*(y - mn)/sd + M[i])
      } else {
        out <- c(out,rep(M[i],N[i]))
      }
    } else out <- c(out, M)
  }
  out
}

### Some patches for nlme; these are relevant for version 3.1-18, but will
### likely be fixed in a future version

getCovariateFormula <- function (object)
{
  form <- formula(object)
  if (!(inherits(form, "formula"))) {
    stop("\"Form\" must be a formula")
  }
  form <- form[[length(form)]]
  if (length(form) == 3 && form[[1]] == as.name("|")) {
    form <- form[[2]]
  }
  ## original version used eval(parse(paste(deparse(form)))), which leads
  ## to nonsense if form is longer than 60 characters
  eval(substitute(~form))
}
logLik.reStruct <- function (object, conLin)
{
  if (any(!is.finite(conLin$Xy)))
    return(-Inf)
  ## original uses the default, NAOK=FALSE; this means .C chokes on Infs,
  ## though the c-language code can handle them just fine (at least on
  ## x86 Linux).  Probably ought to check for ieee arithmetic, if the
  ## patch were for general use.
  .C("mixed_loglik", as.double(conLin$Xy), as.integer(unlist(conLin$dims)),
     as.double(pdFactor(object)),
     as.integer(attr(object, "settings")),
     loglik = double(1), double(1),
     NAOK=TRUE, PACKAGE = "nlme")$loglik
}
```

b. Part 2: Scripts For Data Analysis

i. .Rprofile

R uses the file .Rprofile to set various default conditions before other commands are run. Commands in this file are automatically executed each time R is run from the directory where this file resides.

```
### This loading order assures that the patches included in RBMDS get applied
### to nlme
library(nlme)
library(RBMDS)
```

ii. getalldata.R

This script constructs a single R dataframe from all the various text sources, and does all the error correcting and study deletion in one place. It also creates dotplots of control values by study and unit, to help look for 'units' errors or inconsistencies in the definitions of units among studies (mainly important within chemical). This is run by executing the command (in Unix)

```
R --save < getalldata.R > getalldata.Rout
```

In Windows, R would be replaced by Rterm in the above command.

```
### getalldata.R
### This script reads all the csv files containing OP data (means and sd)
### merges them into a single dataset, and does the data manipulations done
### by GetData (other than dropping small datasets).  Note that all the
### original datasets contain a replicate of the METHAMIDOPHOS data, so those
### must be purged.

oplist <- vector("list",6)
fnames <- c("final5-18part1.csv","final5-18part2.csv",
            "ethopropmethamidophos.csv",
            "oralchlorpyrifosmethylmethamidophos.csv",
            "oraldicrotophosmethamidophs.csv",
            "oralfenthionmethamidophos.csv")
for (i in seq(along=oplist)) oplist[[i]] <- read.csv(fnames[i])

### Now, delete the METHAMIDOPHOS-related records in all but the first
### dataset

for (i in 2:length(oplist)) {
  sel <- oplist[[i]]$CHEMICAL == "METHAMIDOPHOS"
  oplist[[i]] <- oplist[[i]][!sel,]
  oplist[[i]]$CHEMICAL <- factor(oplist[[i]]$CHEMICAL)
}

opdata <- do.call("rbind",oplist)

### Drop "STUDYTYPE" from opdata

opdata <- opdata[,-grep("STUDYTYPE",names(opdata))]

### Rename the variables to what I've been using:
names(opdata) <- c("chemical", "mrid", "guideline","compartment", "sex",
                   "time", "timeunit",  "dose", "chei", "chunit", "sd", "n",
                   "duplicate")
opdata$RowNum <- seq(1:nrow(opdata))

tmpdose <- as.numeric(as.character(opdata$dose))
tmpchei <- as.numeric(as.character(opdata$chei))
tmpn <- as.numeric(as.character(opdata$n))
tmpsd <- as.numeric(as.character(opdata$sd))

sel <- is.na(tmpdose) | is.na(tmpchei) | is.na(tmpn) | is.na(tmpsd)
if (any(sel)) {
  opdata$dose <- tmpdose
  opdata$chei <- tmpchei
  opdata$n <- tmpn
  opdata$sd <- tmpsd
  opdata <- opdata[!sel,]
  }
```

```
### Some data need to be dropped:
### mrid == 43088601, MEVINPHOS
###         41886301, DIAZINON
###         44189501, OXYDEMETONMETHYL
###            28464, CHLORPYRIFOS METHYL
###           116408, FENTHION
###            75810, FENTHION
###           125233, PHORATE

sel <- (opdata$chemical == "MEVINPHOS" & opdata$mrid == 43088601) |
       (opdata$chemical == "DIAZINON" & opdata$mrid == 41886301) |
       (opdata$chemical == "OXYDEMETONMETHYL" & opdata$mrid == 44189501) |
       (opdata$chemical == "CHLORPYRIPHOSMETHYL" & opdata$mrid == 28464) |
       (opdata$chemical == "CHLORPYRIPHOSMETHYL" & opdata$mrid == 29504) |
       (opdata$chemical == "FENTHION" & opdata$mrid == 116408) |
       (opdata$chemical == "FENTHION" & opdata$mrid == 75810) |
       (opdata$chemical == "PHORATE" & opdata$mrid == 125233)

opdata <- opdata[!sel,]

### Change MRID No. 41850002 to 146873

opdata$mrid[opdata$mrid == 41850002] <- 146873

opdata$block <- factor(paste(as.character(opdata$chemical),
                             as.character(opdata$mrid),
                             as.character(opdata$compartment),
                             as.character(opdata$sex),
                             as.character(opdata$time),
                             as.character(opdata$duplicate),
                             sep="-"))

opdata$chemical <- factor(opdata$chemical)
opdata$compartment <- factor(opdata$compartment)
opdata$sex <- factor(opdata$sex)
opdata$duplicate <- factor(opdata$duplicate)
opdata$block <- factor(opdata$block)

### Fixup records with mean=0 and sd=NA, with a warning:

sel <- !is.na(opdata$chei) & opdata$chei == 0 & is.na(opdata$sd)
if (any(sel)) {
  opdata$sd[sel] <- 0
  cat("The following records have had their sds changed from NA to 0\n")
  print(opdata[sel,c("chemical","compartment","sex","time","duplicate",
                     "dose")])
}

sel <- is.na(opdata$chei) | is.na(opdata$sd)
if (any(sel)) {
  cat("The following records have missing values for chei or sd, and will be
dropped\n")

print(opdata[sel,c("chemical","compartment","sex","time","duplicate","dose","chei","sd
")])
  opdata <- opdata[!sel,]
}

### Revise the doses re:
newdosesfile <- "reviseddosesoct10.v2.CSV"

newdoses <- read.csv(newdosesfile)

opdata$olddose <- opdata$dose

### In Mevinphos males in mrid 42588501, there are two groups at 49 days
### with 1 mg/kg/day (nominal), with two actual average dose rates.  This is
### because the second group was originally dosed at 1.5 mg/kg/day, but was
### switched to 1 mg/kg/day at 36 days, due to mortality.  Thus, there are
### two actual doses for the nominal dosage of 1 mg/kg/day.  On day 49, the
### records that occur earlier in the dataset were the original 1 mg/kg/day
### group, and their actual dosage should be 1.12; those that occur next were
### originally 1.5, and should be 1.67.  On day 91, they should all be 1.12.

hash <- paste(opdata$chemical,opdata$mrid,opdata$sex,opdata$olddose,sep=":")

sel91 <- hash == "MEVINPHOS:42588501:M:1" & opdata$time == 91

hash[sel91] <- paste(hash[sel91],"LOWER",sep=":")

### The following won't work if the data are rearranged!

sel49 <- hash == "MEVINPHOS:42588501:M:1" & opdata$time == 49

hash[sel49] <- paste(hash[sel49],rep(c("LOWER","HIGHER"),2),sep=":")

rnd <- paste(newdoses$CHEMICAL,newdoses$MRIDNO.,newdoses$SEX,newdoses$OLDDOSE,sep=":")

sel <- rnd == "MEVINPHOS:42588501:M:1"
rnd[sel] <- paste(rnd[sel],c("LOWER","HIGHER"),sep=":")

rownames(newdoses) <- rnd

Doses <- opdata$olddose

for (i in rnd) {
  Doses[hash == i] <- newdoses[i,"NEWDOSE"]
}

opdata$dose <- as.vector(Doses)

### We are going to make dotcharts of the control activities by mrid
### for each separate unit, before and after the following changes.

require(grid)
require(lattice)

trellis.device("postscript", color=F, file="ControlValsBefore.ps")
BRAINCdata <- opdata[opdata$compartment == "BRAIN" &
                     opdata$dose == 0 &
                     opdata$duplicate %in% c("WHOLE","DUPLICATEWHOLE",
                                             "WHOLEDUPLICATE"),]
BRAINCdata$chemical <- factor(BRAINCdata$chemical)
BRAINCdata$chunit <- factor(BRAINCdata$chunit, exclude=NULL)
levels(BRAINCdata$chunit)[grep("NA",levels(BRAINCdata$chunit))] <- "Unk"

for (Un in levels(BRAINCdata$chunit)) {
  sel <- BRAINCdata$chunit == Un
  print(dotplot(factor(mrid) ~ chei, data=BRAINCdata[sel,],
                main=paste("Before Transformation:",Un)))
}
dev.off()



### Fix some units:

levels(opdata$chunit)[grep("U\\\\L",levels(opdata$chunit))] <- "U/L"
tmp <- as.character(opdata$chunit)
tmp[opdata$mrid %in% c(44895301,44895302) & opdata$compartment == "BRAIN"] <- "U/L"

## Ethoprop - hopefully a temporary fix
tmp[opdata$mrid == 40291801 & opdata$compartment == "BRAIN" & opdata$chemical ==
"ETHOPROP"] <- "U/G1"
tmp[opdata$mrid == 138636 & opdata$compartment == "BRAIN" & opdata$chemical ==
"ETHOPROP"] <- "U/G2"
```

```
tmp[opdata$mrid == 42530201 & opdata$compartment == "BRAIN" & opdata$chemical ==
"ETHOPROP"] <- "U/G3"

## Dicrotophos: one U/L should be U/G
tmp[opdata$mrid==44527802 & opdata$compartment == "BRAIN" & opdata$chemical ==
"DICROTOPHOS" & opdata$chunit == "U/L"] <- "U/G"

## Terbufos: a U/G should be U/mL

tmp[opdata$mrid==44842302 & opdata$compartment == "BRAIN" & opdata$chemical ==
"TERBUFOS"] <- "U/mL"

## Phorate: two mrids original units were U/mL; these were converted in the
## dataset to U/L by multiplying by 1000.  The resulting AChE activities are
## about 10-fold higher than the other U/L measurements in other mrids.  This
## messes with the model fits.  So, divide these measurements by 1000, and
## convert units back to U/mL.

sel <- opdata$mrid %in% c(44895301,44895302)  & opdata$chemical == "PHORATE" &
        opdata$compartment == "BRAIN"
tmp[sel] <- "U/mL"
opdata$chei[sel] <- opdata$chei[sel]/1000
opdata$sd[sel] <- opdata$sd[sel]/1000

opdata$chunit <- factor(tmp)

trellis.device("postscript", color=F, file="ControlValsAfter.ps")
BRAINCdata <- opdata[opdata$compartment == "BRAIN" &
                     opdata$dose == 0 &
                     opdata$duplicate %in% c("WHOLE","DUPLICATEWHOLE",
                                             "WHOLEDUPLICATE"),]
BRAINCdata$chemical <- factor(BRAINCdata$chemical)
BRAINCdata$chunit <- factor(BRAINCdata$chunit, exclude=NULL)
levels(BRAINCdata$chunit)[grep("NA",levels(BRAINCdata$chunit))] <- "Unk"

for (Un in levels(BRAINCdata$chunit)) {
  sel <- BRAINCdata$chunit == Un
  print(dotplot(factor(mrid) ~ chei, data=BRAINCdata[sel,],
                main=paste("After Transformation:",Un)))
}
dev.off()

save.image()
```

iii. BrainProfLik.R

Calculates profile likelihoods for *tB*; the result for each chemical is stored in the file BRAIN-Models-BPL/chem, where chem is the name of the current chemical. The results can be retrieved with ‘load(file.path(“BRAIN-Models-BPL”,chem))’, and will then be in the object ‘xx’. This script needs to be invoked at the command line thus:

```
R –no-save SimStart=1 SimStop=12 < BrainProfLik.R > BrainProfLik.Rout
```

where 1 and 12 are indexes into the list of chemicals; SimStart must be less than or equal to SimStop, and both must be less than 29. Thus, on a cluster of machines with a shared file system, you can start multiple instances of the script.

```
### BrainProfLik.R
### Profile likelihoods for exponential model and male,female values of B
###
### Get the command line arguments (to specify which chemicals to run):

cmdargs <- commandArgs()
if (length(grep("SimStart",cmdargs))==0 || length(grep("SimStop",cmdargs))==0)
  stop("Need arguments SimStart AND SimStop")
SimStart <- as.numeric(strsplit(cmdargs[grep("SimStart",cmdargs)],"=")[[1]][2])
SimStop <- as.numeric(strsplit(cmdargs[grep("SimStop",cmdargs)],"=")[[1]][2])

if (!is.finite(SimStart) || !is.finite(SimStop) || SimStart > SimStop)
  stop("SimStart or SimStop not correctly specified")

### Create a BRAIN only set from opdata

BRAINdata <- opdata[opdata$compartment == "BRAIN"&
                    opdata$duplicate %in% c("WHOLE","DUPLICATEWHOLE",
                                            "WHOLEDUPLICATE"),]

BRAINdata$chemical <- factor(BRAINdata$chemical)
BRAINdata$chunit <- factor(BRAINdata$chunit, exclude=NULL)
levels(BRAINdata$chunit)[grep("NA",levels(BRAINdata$chunit))] <- "Unk"

### Create the grid of points for the profile likelihood:

Bgrid <- expand.grid(B.F=seq(0.01,0.99,length=11),
                     B.M=seq(0.01, 0.99, length=11))

Chemicals <- sort(levels(BRAINdata$chemical))

for (chem in Chemicals[SimStart:SimStop]) {

  cat(paste("-------\n",chem,"\n\n"))
    if (file.exists(file.path("BRAIN-Models-BPL",chem)) &&
        file.info(file.path("BRAIN-Models-BPL",chem))[1,"size"] > 4*1024) {
      cat("Already done\n")
      next
    }

#### Construct the data set

  sel <- BRAINdata$chemical == chem
  seldata <- BRAINdata[sel,]
  seldata$block <- factor(seldata$block)
  seldata$mrid <- factor(seldata$mrid)
```

```
### loop through seldata (by block), using PhonyDF to expand it to
### synthetic individual data.


  tmp <- by(seldata,list(seldata$block),function(x) {
    PhonyDF(x$dose,x$n,x$chei,x$sd,"Dose","AChE",
            chem=rep(as.character(x$chunit[1]),nrow(x)),
            ChemName="Unit")
  })

  for (i in seq(along=tmp)) {
    flds <- unlist(strsplit(names(tmp[i]),"-"))
    N <- nrow(tmp[[i]])
    tmp[[i]]$mrid <- rep(flds[2],N)
    tmp[[i]]$sex <- rep(flds[4],N)
    tmp[[i]]$set <- rep(paste(flds[2],flds[5],flds[6],sep=":"),N)
  }

  Pseldata <- do.call("rbind",unclass(tmp))
  row.names(Pseldata) <- seq(along=Pseldata[[1]])
  Pseldata$set <- factor(Pseldata$set)
  Respscale <- max(seldata$chei)
  Dosescale <- max(seldata$dose)

  Pseldata$AChE.scaled <- Pseldata$AChE/Respscale
  Pseldata$Dose.scaled <- Pseldata$Dose/Dosescale
  Pseldata$s.U <- factor(interaction(Pseldata$sex, Pseldata$Unit, drop=TRUE))

#### Try to estimate exponential model with A, B, and m

### Use the old approach to getting initial values.

  ## Get starting values for A ~ sex:Units - 1
  controldata <- Pseldata[Pseldata$Dose == 0,]
  controldata$lAChE.scaled <- log(controldata$AChE.scaled)
  Astart.lme <- try(lme(lAChE.scaled ~ s.U - 1, data=controldata,
                        random=~1|set))
  if (inherits(Astart.lme,"try-error")) {

    Astart.lm <- lm(lAChE.scaled ~ s.U - 1, data=controldata)
    Astart.lme <- list(coefficients=list(fixed=coef(Astart.lm)))
  }
  ## Get starting values for m
  mstart.F <- as.vector(mean(mhats[grep(paste(chem,"BRAIN F WHOLE"),
                                        rownames(mhats)),1]) +
                        log(Dosescale))
  mstart.M <- as.vector(mean(mhats[grep(paste(chem,"BRAIN M WHOLE"),
                                        rownames(mhats)),1]) +
                        log(Dosescale))

  start <- c(Astart.lme$coefficients$fixed,
             mstart.F,
             mstart.M
             )

### Set up the random effects

### set up the random component:  These are the possibilities:
###
### one mrid, one unit, multiple sets:
###    RnDoM <- list(A+B+m ~ 1| set)
### multiple mrids, one mrid/unit (#mrids == #units), multiple sets:
###    RnDoM <- list(A ~ 1 | set, B+m ~ 1 | mrid/set
### multiple mrids, one mrid/unit (#mrids == #units), one set/mrid:
###    RnDoM <- list(B+m ~ 1 | mrid)
### multiple mrids, multiple mrids/unit (#mrids > #units), multiple sets:
###    RnDoM <- list(A+B+m ~ 1 | mrid/set)
### multiple mrids, multiple mrids/unit ($mrids > #units), one set/mrid:
```

```
###    RnDoM <- list(A+B+m ~ 1 | mrid)

  if (length(levels(Pseldata$mrid)) == 1 &&
      length(levels(Pseldata$set)) == 1) {
    RnDoM <- NULL
    RModel <- NULL
  }
  if (length(levels(Pseldata$mrid)) == 1 && length(levels(Pseldata$set)) > 1) {
    RnDoM <- list(set=pdDiag(form=A+m ~ 1))
    RModel <- A+m~1 | set
  }
  if (length(levels(Pseldata$mrid)) > 1) {
    if (length(levels(Pseldata$mrid)) == length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) > length(levels(Pseldata$mrid))) {
      RnDoM <- list(mrid=pdDiag(form=m ~ 1),
                    set=pdDiag(form=m~1))
      RModel <- m~1|mrid/set
    }
    if (length(levels(Pseldata$mrid)) == length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) == length(levels(Pseldata$mrid))) {
      RnDoM <- list(mrid=pdDiag(form=m ~ 1))
      RModel <- m~1|mrid
    }
    if (length(levels(Pseldata$mrid)) > length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) > length(levels(Pseldata$mrid))) {
      RnDoM <- list(mrid=pdDiag(form=A+m~1),
                    set=pdDiag(form=A+m~1))
      RModel <- A+m~1|mrid/set
    }
    if (length(levels(Pseldata$mrid)) > length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) == length(levels(Pseldata$mrid))) {
      RnDoM <- list(mrid=pdDiag(form=A+m ~ 1))
      RModel <- A+m~1|mrid
    }
  }
### Try fixing B
### to a range of B's from 0.01 to 0.99, and saving the resulting 'LL'
### from nlme or gnls in a vector.
  LLs <- vector("numeric",nrow(Bgrid))

  for (i in seq(along=LLs)) {
    cat(paste("\n------\n",i,":",Bgrid$B.F[i],Bgrid$B.M[i],"\n\n"))
    Model <- eval(substitute(AChE.scaled ~ CexpBS(Dose.scaled,A,m,sex,
                                                  fixed=x),
                             list(x=list(B=c(F=as.vector(
                                                log(Bgrid$B.F[i]/
                                                    (1 - Bgrid$B.F[i]))),
                                            M=as.vector(
                                              log(Bgrid$B.M[i]/
                                                  (1 - Bgrid$B.M[i]))))))))


    if (!is.null(RnDoM)) {
      fit <-
        try(eval(substitute(nlme(Model,data=Pseldata,
                                 fixed=list(A ~ s.U - 1, m ~ sex-1),
                                 random=xxxx,
                                 start=start,
                                 weights=varPower(fixed=0.5),
                                 method="ML"),
                            list(Model=Model,xxxx=RnDoM))))
    } else {
      fit <-
        try(eval(substitute(gnls(Model, data=Pseldata,
                                 params=list(A ~ s.U - 1, m ~ sex-1),
                                 start=start,
                                 weights=varPower(fixed=0.5)),
                            list(Model=Model))))
    }
    LLs[i] <- if (!inherits(fit, "try-error")) logLik(fit) else NA
    xx <- list(Chemical=chem, Model=Model, Start=start,
               Respscale=Respscale,Dosescale=Dosescale, Random=RModel,
               Data=seldata,Bgrid=Bgrid,LL=LLs)
    save(xx,file=file.path("BRAIN-Models-BPL",chem))
  }
}
```

iv. plotProfiles.R

Plots the profile likelihoods computed by the previous script for each chemical in a pdf file. Can be invoked as:

```
R --no-save < plotProfiles.R > plotProfiles.Rout
```

```
### plotProfiles.R
require(akima)
pdf(file="Profiles4B.pdf")
dirname <- "BRAIN-Models-BPL"
Modeldirname <- "BRAIN-Models-1"
for (chem in Chemicals){
  if (!file.exists(file.path(dirname,chem))) next
  load(file.path(dirname,chem))
  Bgrid <- xx$Bgrid
  Bgrid$LL <- xx$LL
### Ethoprop inexplicably dies with an out-of-memory error before it is done
  if (chem == "ETHOPROP")Bgrid$LL[Bgrid$LL == 0] <- NA
  Bgrid <- na.omit(Bgrid)
  out <- interp(Bgrid$B.F,Bgrid$B.M,Bgrid$LL,seq(0,1,length=40),
                seq(0,1,length=40))
  image(out,xlab="B.F",ylab="B.M",main=chem)
  points(Bgrid$B.F,Bgrid$B.M,pch=3,cex=0.5)

  if (file.exists(file.path(Modeldirname,chem)) &&
      file.info(file.path(Modeldirname,chem))[1,"size"] > 3*1024) {
    load(file.path(Modeldirname,chem))
    if (inherits(xx$Fit,"nlme")) {
      B.F <- xx$Fit$coefficients$fixed["B.sexF"]
      B.M <- xx$Fit$coefficients$fixed["B.sexM"]
    } else {
      B.F <- xx$Fit$coefficients["B.sexF"]
      B.M <- xx$Fit$coefficients["B.sexM"]
    }
    B.F <- 1/(1 + exp(-B.F))
    B.M <- 1/(1 + exp(-B.M))
    points(B.F,B.M)
  }
}
dev.off()
```

v. plotProfiles2.R

This is an interactive version of the previous code, designed to facilitate identifying the peak of the profile likelihood. This must be invoked from within a running R process using

```
source(“plotProfiles2.R”)
```

```
### plotProfiles2.R
require(akima)
dirname <- "BRAIN-Models-BPL"
Modeldirname <- "BRAIN-Models-1"
Bstart2 <- data.frame(B.F=vector("numeric",length(Chemicals)),
                      B.M=vector("numeric",length(Chemicals)),
                      row.names=Chemicals)
for (chem in Chemicals){
  if (!file.exists(file.path(dirname,chem))) next
  load(file.path(dirname,chem))
  Bgrid <- xx$Bgrid
  Bgrid$LL <- xx$LL
### Ethoprop inexplicably dies with an out-of-memory error before it is done
  if (chem == "ETHOPROP")Bgrid$LL[Bgrid$LL == 0] <- NA
  Bgrid <- na.omit(Bgrid)
  out <- interp(Bgrid$B.F,Bgrid$B.M,Bgrid$LL,seq(0,1,length=40),
                seq(0,1,length=40))
  image(out,xlab="B.F",ylab="B.M",main=chem)
  points(Bgrid$B.F,Bgrid$B.M,pch=3,cex=0.5)

  if (file.exists(file.path(Modeldirname,chem)) &&
      file.info(file.path(Modeldirname,chem))[1,"size"] > 3*1024) {
    load(file.path(Modeldirname,chem))
    if (inherits(xx$Fit,"nlme")) {
      B.F <- xx$Fit$coefficients$fixed["B.sexF"]
      B.M <- xx$Fit$coefficients$fixed["B.sexM"]
    } else {
      B.F <- xx$Fit$coefficients["B.sexF"]
      B.M <- xx$Fit$coefficients["B.sexM"]
    }
    B.F <- 1/(1 + exp(-B.F))
    B.M <- 1/(1 + exp(-B.M))
    points(B.F,B.M)
  }
  tmp <- locator(1)
  Bstart2[chem,"B.F"] <- tmp$x
  Bstart2[chem,"B.M"] <- tmp$y
}
save.image()
```

vi. BRAINfits2.R

This script actually fits the models. It is invoked as is BrainProfLik.R, assigning values to SimStart and SimStop.

```
### BRAINfits2.R
### Fit the basic modified exponential model to all the Brain data
### using nlme when there are nested effects, and gnls otherwise.
###
### Get the command line arguments (to specify which chemicals to run):

cmdargs <- commandArgs()
if (length(grep("SimStart",cmdargs))==0 || length(grep("SimStop",cmdargs))==0)
  stop("Need arguments SimStart AND SimStop")
SimStart <- as.numeric(strsplit(cmdargs[grep("SimStart",cmdargs)],"=")[[1]][2])
SimStop <- as.numeric(strsplit(cmdargs[grep("SimStop",cmdargs)],"=")[[1]][2])

if (!is.finite(SimStart) || !is.finite(SimStop) || SimStart > SimStop)
  stop("SimStart or SimStop not correctly specified")

### Create a BRAIN only set from opdata

BRAINdata <- opdata[opdata$compartment == "BRAIN"&
                    opdata$duplicate %in% c("WHOLE","DUPLICATEWHOLE",
                                            "WHOLEDUPLICATE"),]

BRAINdata$chemical <- factor(BRAINdata$chemical)
BRAINdata$chunit <- factor(BRAINdata$chunit, exclude=NULL)
levels(BRAINdata$chunit)[grep("NA",levels(BRAINdata$chunit))] <- "Unk"

Chemicals <- sort(levels(BRAINdata$chemical))

for (chem in Chemicals[SimStart:SimStop]) {

  cat(paste("-------\n",chem,"\n\n"))
    if (file.exists(file.path("BRAIN-Models-2-save",chem)) &&
        file.info(file.path("BRAIN-Models-2-save",chem))[1,"size"] > 15*1024) {
      cat("Already done\n")
      next
    }

#### Construct the data set

  sel <- BRAINdata$chemical == chem
  seldata <- BRAINdata[sel,]
  seldata$block <- factor(seldata$block)
  seldata$mrid <- factor(seldata$mrid)

### loop through seldata (by block), using PhonyDF to expand it to
### synthetic individual data.


  tmp <- by(seldata,list(seldata$block),function(x) {
    PhonyDF(x$dose,x$n,x$chei,x$sd,"Dose","AChE",
            chem=rep(as.character(x$chunit[1]),nrow(x)),
            ChemName="Unit")
  })

  for (i in seq(along=tmp)) {
    flds <- unlist(strsplit(names(tmp[i]),"-"))
    N <- nrow(tmp[[i]])
    tmp[[i]]$mrid <- rep(flds[2],N)
    tmp[[i]]$sex <- rep(flds[4],N)
    tmp[[i]]$set <- rep(paste(flds[2],flds[5],flds[6],sep=":"),N)
  }
  Pseldata <- do.call("rbind",unclass(tmp))
  row.names(Pseldata) <- seq(along=Pseldata[[1]])
  Pseldata$set <- factor(Pseldata$set)
  Respscale <- max(seldata$chei)
  Dosescale <- max(seldata$dose)

  Pseldata$AChE.scaled <- Pseldata$AChE/Respscale
  Pseldata$Dose.scaled <- Pseldata$Dose/Dosescale
  Pseldata$s.U <- factor(interaction(Pseldata$sex, Pseldata$Unit, drop=TRUE))

#### Get starting values for A ~ sex:Units - 1
  controldata <- Pseldata[Pseldata$Dose == 0,]
  controldata$lAChE.scaled <- log(controldata$AChE.scaled)
  Astart.lme <- try(lme(lAChE.scaled ~ s.U - 1, data=controldata,
                        random=~1|set))
  if (inherits(Astart.lme,"try-error")) {

    Astart.lm <- lm(lAChE.scaled ~ s.U - 1, data=controldata)
    Astart.lme <- list(coefficients=list(fixed=coef(Astart.lm)))
  }
#### Get starting values for B
  Bs.F <- Bstart2[chem,"B.F"]
  Bs.F <- as.vector(log(Bs.F/(1 - Bs.F)))
  Bs.M <- Bstart2[chem,"B.M"]
  Bs.M <- as.vector(log(Bs.M/(1 - Bs.M)))

### values for A and m
### set up the random component:  These are the possibilities:
###
### one mrid, one unit, multiple sets:
###    RnDoM <- list(A+m ~ 1| set)
### multiple mrids, one mrid/unit (#mrids == #units), multiple sets:
###    RnDoM <- list(A ~ 1 | set, m ~ 1 | mrid/set
### multiple mrids, one mrid/unit (#mrids == #units), one set/mrid:
###    RnDoM <- list(m ~ 1 | mrid)
### multiple mrids, multiple mrids/unit (#mrids > #units), multiple sets:
###    RnDoM <- list(A+m ~ 1 | mrid/set)
### multiple mrids, multiple mrids/unit ($mrids > #units), one set/mrid:
###    RnDoM <- list(A+m ~ 1 | mrid)

  if (length(levels(Pseldata$mrid)) == 1 &&
      length(levels(Pseldata$set)) == 1) {
    RnDoM1 <- NULL
    RModel1 <- NULL
    RnDoM <- NULL
    RModel <- NULL
  }
  if (length(levels(Pseldata$mrid)) == 1 && length(levels(Pseldata$set)) > 1) {
    RnDoM1 <- list(set=pdDiag(form=A+m ~ 1))
    RModel1 <- A+m~1 | set
    RnDoM <- list(set=pdDiag(form=A+B+m ~ 1))
    RModel <- A+B+M~1 | set
  }
  if (length(levels(Pseldata$mrid)) > 1) {
    if (length(levels(Pseldata$mrid)) == length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) > length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=m ~ 1),
                     set=pdDiag(form=m~1))
      RModel1 <- m~1|mrid/set
      RnDoM <- list(mrid=pdDiag(form=B+m ~ 1),
                     set=pdDiag(form=B+m~1))
      RModel <- B+m~1|mrid/set
    }
    if (length(levels(Pseldata$mrid)) == length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) == length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=m ~ 1))
      RModel1 <- m~1|mrid
      RnDoM <- list(mrid=pdDiag(form=B+m ~ 1))
      RModel <- B+m~1|mrid
    }
    if (length(levels(Pseldata$mrid)) > length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) > length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=A+m~1),
                     set=pdDiag(form=A+m~1))
      RModel1 <- A+m~1|mrid/set
      RnDoM <- list(mrid=pdDiag(form=A+B+m~1),
                     set=pdDiag(form=A+B+m~1))
      RModel <- A+B+m~1|mrid/set
    }
    if (length(levels(Pseldata$mrid)) > length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) == length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=A+m ~ 1))
      RModel1 <- A+m~1|mrid
      RnDoM <- list(mrid=pdDiag(form=A+B+m ~ 1))
      RModel <- A+B+m~1|mrid
    }
  }

### Fix B at the starting values and estimate m to get revised 'start'

  Model <- eval(substitute(AChE.scaled ~ CexpBS(Dose.scaled,A,m,sex,
                                               fixed=list(B=c(F=xxxx,
                                                              M=yyyy))),
                           list(xxxx=Bs.F,yyyy=Bs.M)))
  mstart.F <- as.vector(mean(mhats[grep(paste(chem,"BRAIN F WHOLE"),
                                        rownames(mhats)),1]) +
                        log(Dosescale))
  mstart.M <- as.vector(mean(mhats[grep(paste(chem,"BRAIN M WHOLE"),
                                        rownames(mhats)),1]) +
                        log(Dosescale))

  start <- c(Astart.lme$coefficients$fixed,
             m.sexF=mstart.F,
             m.sexM=mstart.M
             )
  if (!is.null(RnDoM1)) {
    fitNoB <-
      try(eval(substitute(nlme(Model,data=Pseldata,
                               fixed=list(A ~ s.U - 1, m ~ sex-1),
                               random=xxxx,
                               start=start,
                               weights=varPower(fixed=0.5),
                               method="ML",verbose=TRUE),
                          list(Model=Model,xxxx=RnDoM1))))
  } else {
    fitNoB <-
      try(eval(substitute(gnls(Model, data=Pseldata,
                               params=list(A ~ s.U - 1, m ~ sex-1),
                               start=start,
                               weights=varPower(fixed=0.5),
                               verbose=TRUE),
                          list(Model=Model))))
  }

### Get a new 'start' vector from the coefficient vector of fitNoB
### Since it is based on previous fits, I'm assuming the fit was successful.
### This may be dangerous, but I don't think so.
  if (inherits(fitNoB,"try-error")) {
    cat("Even fixed B fit doesn't work!\n")
    next
  }

  start <- if (inherits(fitNoB,"nlme")) fitNoB$coefficients$fixed else
               fitNoB$coefficients
  start <- c(start[grep("^A",names(start))],
             B.sexF = Bs.F,
             B.sexM = Bs.M,
             start[c("m.sexF","m.sexM")])
### Set up the model

  Model <- AChE.scaled ~ CexpB(Dose.scaled,A,B,m)
  ## Try a few times to get a good fit before giving up, tweaking the start
  ## vector each time
  Tstart <- start
  trycount <- 0
  trymax <- 10
  repeat {
    trycount <- trycount + 1
    if (!is.null(RnDoM)) {
      fitWB <-
        try(eval(substitute(nlme(Model,data=Pseldata,
                                 fixed=list(A ~ s.U - 1, B ~ sex - 1,
                                   m ~ sex-1),
                                 random=xxxx,
                                 start=Tstart,
                                 weights=varPower(fixed=0.5),
                                 method="ML",verbose=TRUE),
                            list(Model=Model,xxxx=RnDoM))))
    } else {
      fitWB <-
        try(eval(substitute(gnls(Model, data=Pseldata,
                                 params=list(A ~ s.U - 1, B ~ sex-1,
                                   m ~ sex-1),
                                 start=Tstart,
                                 weights=varPower(fixed=0.5),
                                 verbose=TRUE),
                            list(Model=Model))))
    }
    if (!inherits(fitWB,"try-error") || trycount >= trymax) break
    Tstart <- start * runif(length(start),min=0.9,max=1.1)
  }

  EstB <- !inherits(fitWB,"try-error")

### Next, the pk extension to CexpB: CpkexpB
  if (EstB) {
    ## Get start values from fitWB
    start <- if (inherits(fitWB, "nlme")) {
      fitWB$coefficients$fixed
    } else {
      fitWB$coefficients
    }
    ## need to add S and D to the end.
    start["S"] <- log(0.0001)
    start["D"] <- log(0.2)
    ## set up the model
    Model <- AChE.scaled ~ CpkexpB(Dose.scaled,A,B,m,S,D)
    ## estimate it
    ## Try a few times to get a good fit before giving up; tweaking the
    ## start vector each time
    Tstart <- start
    trycount <- 0
    trymax <- 10
    repeat {
      trycount <- trycount + 1
      if (!is.null(RnDoM)) {
        fitpk <-
          try(eval(substitute(nlme(Model,data=Pseldata,
                                   fixed=list(A ~ s.U - 1, B ~ sex - 1,
                                     m ~ sex - 1, S ~ 1, D ~ 1),
                                   random=xxxx,
                                   start=zzzz,
                                   weights=varPower(fixed=0.5),
                                   method="ML",
                                   verbose=TRUE),
                              list(Model=Model,xxxx=RnDoM,
                                   zzzz=Tstart))))
      } else {
        fitpk <-
          try(eval(substitute(gnls(Model,data=Pseldata,
                                   params=list(A ~ s.U - 1, B ~ sex - 1,
                                     m ~ sex - 1, S ~ 1, D ~ 1),
                                   start=zzzz,
                                   weights=varPower(fixed=0.5),
                                   verbose=TRUE),
                              list(Model=Model,
                                   zzzz=Tstart))))
      }
      if (!inherits(fitpk,"try-error") || trycount >= trymax) break
      Tstart <- start * runif(length(start),min=0.9,max=1.1)
    }
  } else {
    ## Get start values from fitNoB
    start <- if (inherits(fitNoB, "nlme")) fitNoB$coefficients$fixed else
                 fitNoB$coefficients
    ## need to add S and D to the end.
    start["S"] <- log(0.0001)
    start["D"] <- log(0.2)
    ## set up the model
    Model <- eval(substitute(AChE.scaled ~ CpkexpBS(Dose.scaled,A,m,S,D,sex,
                                                    fixed=list(B=c(F=xxxx,
                                                                 M=yyyy))),
                             list(xxxx=Bs.F,yyyy=Bs.M)))
    ## estimate it
    Tstart <- start
    trycount <- 0
    trymax <- 10
    repeat {
      trycount <- trycount + 1
      if (!is.null(RnDoM1)) {
        fitpk <-
          try(eval(substitute(nlme(Model,data=Pseldata,
                                   fixed=list(A ~ s.U - 1, m ~ sex - 1,
                                     S ~ 1, D ~ 1),
                                   random=xxxx,
                                   start=zzzz,
                                   weights=varPower(fixed=0.5),
                                   method="ML",verbose=TRUE),
                              list(Model=Model,xxxx=RnDoM1,
                                   zzzz=Tstart))))
      } else {
        fitpk <-
          try(eval(substitute(gnls(Model, data=Pseldata,
                                   params=list(A ~ s.U - 1, m ~ sex - 1,
                                     S ~ 1, D ~ 1),
                                   start=zzzz,
                                   weights=varPower(fixed=0.5),
                                   verbose=TRUE),
                              list(Model=Model,
                                   zzzz=Tstart))))
      }
      if (!inherits(fitpk,"try-error") || trycount >= trymax) break
      Tstart <- start * runif(length(start), min=0.9, max=1.1)
    }
  }
  ### We may not be able to estimate S, so retry with S fixed to
  ### c(-4,-4)
  if (inherits(fitpk, "try-error")) {
    if (EstB) {
      ## Get start values from fitWB
      start <- if (inherits(fitWB, "nlme")) {
        fitWB$coefficients$fixed
      } else {
        fitWB$coefficients
      }
      ## need to add D to the end.
      start["D"] <- log(0.2)
      ## set up the model
      Model <- AChE.scaled ~ CpkexpBS(Dose.scaled,A,B,m,D,sex,
                                     fixed=list(S=c(F= -8.0, M= -8.0)))
      ## estimate it
      ## Try a few times to get a good fit before giving up; tweaking the
      ## start vector each time
      Tstart <- start
      trycount <- 0
      trymax <- 10
      repeat {
        trycount <- trycount + 1
        if (!is.null(RnDoM)) {
          fitpk <-
            try(eval(substitute(nlme(Model,data=Pseldata,
                                     fixed=list(A ~ s.U - 1, B ~ sex - 1,
                                       m ~ sex - 1, D ~ 1),
                                     random=xxxx,
                                     start=zzzz,
                                     weights=varPower(fixed=0.5),
                                     method="ML",
                                     verbose=TRUE),
                                list(Model=Model,xxxx=RnDoM,
                                     zzzz=Tstart))))
        } else {
          fitpk <-
            try(eval(substitute(gnls(Model,data=Pseldata,
                                     params=list(A ~ s.U - 1, B ~ sex - 1,
                                       m ~ sex - 1, D ~ 1),
                                     start=zzzz,
                                     weights=varPower(fixed=0.5),
                                     verbose=TRUE),
                                list(Model=Model,
                                     zzzz=Tstart))))
        }
        if (!inherits(fitpk,"try-error") || trycount >= trymax) break
        Tstart <- start * runif(length(start),min=0.9,max=1.1)
      }
    } else {
      ## Get start values from fitNoB
      start <- if (inherits(fitNoB, "nlme")) fitNoB$coefficients$fixed else
      fitNoB$coefficients
      ## need to add D to the end.
      start["D"] <- log(0.2)
      ## set up the model
      Model <- eval(substitute(AChE.scaled ~ CpkexpBS(Dose.scaled,A,m,D,sex,
                                                      fixed=list(B=c(F=xxxx,
                                                                   M=yyyy),
                                                        S=c(F=-8.0, M=-8.0))),
                               list(xxxx=Bs.F,yyyy=Bs.M)))
      ## estimate it
      Tstart <- start
      trycount <- 0
      trymax <- 10
      repeat {
        trycount <- trycount + 1
        if (!is.null(RnDoM1)) {
          fitpk <-
            try(eval(substitute(nlme(Model,data=Pseldata,
                                     fixed=list(A ~ s.U - 1, m ~ sex - 1,
                                       D ~ 1),
                                     random=xxxx,
                                     start=zzzz,
                                     weights=varPower(fixed=0.5),
                                     method="ML",verbose=TRUE),
                                list(Model=Model,xxxx=RnDoM1,
                                     zzzz=Tstart))))
        } else {
          fitpk <-
            try(eval(substitute(gnls(Model, data=Pseldata,
                                     params=list(A ~ s.U - 1, m ~ sex - 1,
                                       D ~ 1),
                                     start=zzzz,
                                     weights=varPower(fixed=0.5),
                                     verbose=TRUE),
                                list(Model=Model,
                                     zzzz=Tstart))))
        }
        if (!inherits(fitpk,"try-error") || trycount >= trymax) break
        Tstart <- start * runif(length(start), min=0.9, max=1.1)
      }
    }

  }

### Next, fit the model with 'BMD' instead of 'm'.  If fitWB worked, use the
### version that estimates 'B', else use the fixed B's guessed from the
### profile likelihoods.

###########################################################################
########################### BMR is set here ###############################
  BMR <- 0.1
###########################################################################
### New Random Effects (now we have BMD instead of m)
  if (length(levels(Pseldata$mrid)) == 1 &&
      length(levels(Pseldata$set)) == 1) {
    RnDoM1 <- NULL
    RModel1 <- NULL
    RnDoM <- NULL
    RModel <- NULL
  }
  if (length(levels(Pseldata$mrid)) == 1 && length(levels(Pseldata$set)) > 1) {
    RnDoM1 <- list(set=pdDiag(form=A+BMD ~ 1))
    RModel1 <- A+BMD~1 | set
    RnDoM <- list(set=pdDiag(form=A+B+BMD ~ 1))
    RModel <- A+B+BMD~1 | set
  }
  if (length(levels(Pseldata$mrid)) > 1) {
    if (length(levels(Pseldata$mrid)) == length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) > length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=BMD ~ 1),
                    set=pdDiag(form=BMD~1))
      RModel1 <- BMD~1|mrid/set
      RnDoM <- list(mrid=pdDiag(form=B+BMD ~ 1),
                    set=pdDiag(form=B+BMD~1))
      RModel <- B+BMD~1|mrid/set
    }
    if (length(levels(Pseldata$mrid)) == length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) == length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=BMD ~ 1))
      RModel1 <- BMD~1|mrid
      RnDoM <- list(mrid=pdDiag(form=B+BMD ~ 1))
      RModel <- B+BMD~1|mrid
    }
    if (length(levels(Pseldata$mrid)) > length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) > length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=A+BMD~1),
                    set=pdDiag(form=A+BMD~1))
      RModel1 <- A+BMD~1|mrid/set
      RnDoM <- list(mrid=pdDiag(form=A+B+BMD~1),
                    set=pdDiag(form=A+B+BMD~1))
      RModel <- A+B+BMD~1|mrid/set
    }
    if (length(levels(Pseldata$mrid)) > length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) == length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=A+BMD ~ 1))
      RModel1 <- A+BMD~1|mrid
      RnDoM <- list(mrid=pdDiag(form=A+B+BMD ~ 1))
      RModel <- A+B+BMD~1|mrid
    }
  }

  if (EstB) {
    ## Get new start vector
    start <- if (inherits(fitWB,"nlme")) {
      fitWB$coefficients$fixed
    } else {
      fitWB$coefficients
    }
    ## this 'start' has log(BMD) instead of log(m)
    PB.F <- 1/(1 + exp(-start["B.sexF"]))
    BMD.F <- log(-log((1 - BMR - PB.F)/(1-PB.F))) - start["m.sexF"]
    PB.M <- 1/(1 + exp(-start["B.sexM"]))
    BMD.M <- log(-log((1 - BMR - PB.M)/(1-PB.M))) - start["m.sexM"]
    start <- c(start[grep("^A",names(start))],start[c("B.sexF","B.sexM")],
                        BMD.sexF=as.vector(BMD.F),
                        BMD.sexM=as.vector(BMD.M))
    ## set up the model
    Model <- eval(substitute(AChE.scaled ~ CexpBwD(Dose.scaled,A,B,BMD,
                                                   BMR=xxxx),
                             list(xxxx=BMR)))
    ## Run it
    ## Try a few times to get a good fit before giving up, tweaking the start
    ## vector each time
    Tstart <- start
    trycount <- 0
    trymax <- 10
    repeat {
      trycount <- trycount + 1
      if (!is.null(RnDoM)) {
        fitwD <-
          try(eval(substitute(nlme(Model,data=Pseldata,
                                   fixed=list(A ~ s.U - 1, B ~ sex - 1,
                                     BMD ~ sex-1),
                                   random=xxxx,
                                   start=Tstart,
                                   weights=varPower(fixed=0.5),
                                   method="ML",verbose=TRUE),
                              list(Model=Model,xxxx=RnDoM))))
      } else {
        fitwD <-
          try(eval(substitute(gnls(Model, data=Pseldata,
                                   params=list(A ~ s.U - 1, B ~ sex-1,
                                     BMD ~ sex-1),
                                   start=Tstart,
                                   weights=varPower(fixed=0.5),
                                   verbose=TRUE),
                              list(Model=Model))))
      }
      if (!inherits(fitwD, "try-error") || trycount >= trymax) break
      Tstart <- start * runif(length(start), min=0.9, max=1.1)
    }
  } else {
    ## Get new start vector
    start <- if (inherits(fitNoB,"nlme")) {
      fitNoB$coefficients$fixed
    } else {
      fitNoB$coefficients
    }
```

```
    PB.F <- 1/(1 + exp(-Bs.F))
    BMD.F <- log(-log((1 - BMR - PB.F)/(1-PB.F))) - start["m.sexF"]
    PB.M <- 1/(1 + exp(-Bs.M))
    BMD.M <- log(-log((1 - BMR - PB.M)/(1-PB.M))) - start["m.sexM"]
    start <- c(start[grep("^A",names(start))],
                        BMD.sexF=as.vector(BMD.F),
                        BMD.sexM=as.vector(BMD.M))

    ## set up the model
    Model <- eval(substitute(AChE.scaled ~ CexpBwDS(Dose.scaled,A,BMD,sex,
                                                  fixed=list(B=c(F=xxxx,
                                                               M=yyyy))),
                             list(xxxx=Bs.F,yyyy=Bs.M)))
    ## Run it
    if (!is.null(RnDoM1)) {
      fitwD <-
        try(eval(substitute(nlme(Model,data=Pseldata,
                                 fixed=list(A ~ s.U - 1,
                                   BMD ~ sex-1),
                                 random=xxxx,
                                 start=start,
                                 weights=varPower(fixed=0.5),
                                 method="ML",verbose=TRUE),
                            list(Model=Model,xxxx=RnDoM1))))
    } else {
      fitwD <-
        try(eval(substitute(gnls(Model, data=Pseldata,
                                 params=list(A ~ s.U - 1,
                                   BMD ~ sex-1),
                                 start=start,
                                 weights=varPower(fixed=0.5),
                                 verbose=TRUE),
                            list(Model=Model))))
    }
  }

  xx <- list(Chemical=chem, Model=Model, Fitm=if (EstB) fitWB else fitNoB,
             FitBMD=fitwD,
             Fitpk=fitpk,
             Respscale=Respscale,Dosescale=Dosescale, Random=RModel,
             Data=seldata)
  save(xx,file=file.path("BRAIN-Models-2-save",chem))
}
```

vii. plotBrainresids2.R

Plots various useful diagnostics for the fits of the basic model to the data. Plots go to a pdf file, so this can be run in batch mode:

```
R ––no-save < plotBrainresids2.R > plotBrainresids2.Rout
```

```
### plotBRAINresids2.R

### Create a BRAIN only set from opdata

BRAINdata <- opdata[opdata$compartment == "BRAIN" &
                    opdata$duplicate %in% c("WHOLE","DUPLICATEWHOLE",
                                            "WHOLEDUPLICATE"),]

BRAINdata$chemical <- factor(BRAINdata$chemical)
BRAINdata$chunit <- factor(BRAINdata$chunit, exclude=NULL)
levels(BRAINdata$chunit)[grep("NA",levels(BRAINdata$chunit))] <- "Unk"
Chemicals <- sort(levels(BRAINdata$chemical))

### Create a data set to hold the m's BMDS, potencies, etc.
nchem <- length(Chemicals)
PotencyFrame1 <- data.frame(B.est=as.logical(rep(NA,nchem)),
                            Random=as.logical(rep(NA,nchem)),
                            m.F=as.numeric(rep(NA,nchem)),
                            lm.F=as.numeric(rep(NA,nchem)),
                            lm.F.se=as.numeric(rep(NA,nchem)),
                            m.F.lcl=as.numeric(rep(NA,nchem)),
                            m.F.ucl=as.numeric(rep(NA,nchem)),
                            m.M=as.numeric(rep(NA,nchem)),
                            lm.M=as.numeric(rep(NA,nchem)),
                            lm.M.se=as.numeric(rep(NA,nchem)),
                            m.M.lcl=as.numeric(rep(NA,nchem)),
                            m.M.ucl=as.numeric(rep(NA,nchem)),
                            RPm.F=as.numeric(rep(NA,nchem)),
                            RPm.F.lcl=as.numeric(rep(NA,nchem)),
                            RPm.F.ucl=as.numeric(rep(NA,nchem)),
                            RPm.M=as.numeric(rep(NA,nchem)),
                            RPm.M.lcl=as.numeric(rep(NA,nchem)),
                            RPm.M.ucl=as.numeric(rep(NA,nchem)),
                            BMD.F=as.numeric(rep(NA,nchem)),
                            lBMD.F=as.numeric(rep(NA,nchem)),
                            lBMD.F.se=as.numeric(rep(NA,nchem)),
                            BMD.F.lcl=as.numeric(rep(NA,nchem)),
                            BMD.F.ucl=as.numeric(rep(NA,nchem)),
                            BMD.M=as.numeric(rep(NA,nchem)),
                            lBMD.M=as.numeric(rep(NA,nchem)),
                            lBMD.M.se=as.numeric(rep(NA,nchem)),
                            BMD.M.lcl=as.numeric(rep(NA,nchem)),
                            BMD.M.ucl=as.numeric(rep(NA,nchem)),
                            RPBMD.F=as.numeric(rep(NA,nchem)),
                            RPBMD.F.lcl=as.numeric(rep(NA,nchem)),
                            RPBMD.F.ucl=as.numeric(rep(NA,nchem)),
                            RPBMD.M=as.numeric(rep(NA,nchem)),
                            RPBMD.M.lcl=as.numeric(rep(NA,nchem)),
                            RPBMD.M.ucl=as.numeric(rep(NA,nchem)),
                            row.names=Chemicals)
PotencyFrame1$B.est <- as.logical(PotencyFrame1$B.est)
PotencyFrame1$Random <- as.logical(PotencyFrame1$Random)



pdf("BRAINresids2.pdf")

for (chem in Chemicals) {
  cat(paste("-------\n",chem,"\n\n"))
  if (!file.exists(file.path("BRAIN-Models-2-save",chem)) ||
      file.info(file.path("BRAIN-Models-2-save",chem))[1,"size"] < 4*1024) {
    cat("No Model Fit\n")
    next
  }
  sel <- BRAINdata$chemical == chem
  seldata <- BRAINdata[sel,]
  seldata$block <- factor(seldata$block)
  seldata$mrid <- factor(seldata$mrid)
  seldata$Unit <- factor(seldata$chunit)
  seldata$s.U <- factor(interaction(seldata$sex, seldata$Unit, drop=TRUE))

### load the model fit

  load(file.path("BRAIN-Models-2-save",chem))

### If Fitm didn't work, nothing else will, either.

  if (inherits(xx$Fitm,"try-error")) next

### Fill the corresponding record for PotencyFrame1

  PotencyFrame1[chem,"B.est"] <- !("fixed" %in% names(xx$Fitm$call$model[[3]]))
  PotencyFrame1[chem,"Random"] <- !is.null(xx$Random)
  tTable <- summary(xx$Fitm)$tTable
  PotencyFrame1[chem,"lm.F"] <- tTable["m.sexF",1] - log(xx$Dosescale)
  PotencyFrame1[chem,"lm.M"] <- tTable["m.sexM",1] - log(xx$Dosescale)
  PotencyFrame1[chem,"lm.F.se"] <- tTable["m.sexF",2]
  PotencyFrame1[chem,"lm.M.se"] <- tTable["m.sexM",2]
  if (inherits(xx$Fitm,"nlme")) {
    PotencyFrame1[chem,c("m.F.lcl","m.F","m.F.ucl")] <-
      exp(intervals(xx$Fitm,which="fixed")[[1]]["m.sexF",])/xx$Dosescale
    PotencyFrame1[chem,c("m.M.lcl","m.M","m.M.ucl")] <-
      exp(intervals(xx$Fitm,which="fixed")[[1]]["m.sexM",])/xx$Dosescale
  } else {
    PotencyFrame1[chem,c("m.F.lcl","m.F","m.F.ucl")] <-
      exp(intervals(xx$Fitm,which="coef")[[1]]["m.sexF",])/xx$Dosescale
    PotencyFrame1[chem,c("m.M.lcl","m.M","m.M.ucl")] <-
      exp(intervals(xx$Fitm,which="coef")[[1]]["m.sexM",])/xx$Dosescale
  }
  if (!inherits(xx$FitBMD,"try-error")) {
    tTable <- summary(xx$FitBMD)$tTable
    PotencyFrame1[chem,"lBMD.F"] <- tTable["BMD.sexF",1] + log(xx$Dosescale)
    PotencyFrame1[chem,"lBMD.M"] <- tTable["BMD.sexM",1] + log(xx$Dosescale)
    PotencyFrame1[chem,"lBMD.F.se"] <- tTable["BMD.sexF",2]
    PotencyFrame1[chem,"lBMD.M.se"] <- tTable["BMD.sexM",2]
    if (inherits(xx$FitBMD,"nlme")) {
      PotencyFrame1[chem,c("BMD.F.lcl","BMD.F","BMD.F.ucl")] <-
        exp(intervals(xx$FitBMD,which="fixed")[[1]]["BMD.sexF",])*xx$Dosescale
      PotencyFrame1[chem,c("BMD.M.lcl","BMD.M","BMD.M.ucl")] <-
        exp(intervals(xx$FitBMD,which="fixed")[[1]]["BMD.sexM",])*xx$Dosescale
    } else {
      PotencyFrame1[chem,c("BMD.F.lcl","BMD.F","BMD.F.ucl")] <-
        exp(intervals(xx$FitBMD,which="coef")[[1]]["BMD.sexF",])*xx$Dosescale
      PotencyFrame1[chem,c("BMD.M.lcl","BMD.M","BMD.M.ucl")] <-
        exp(intervals(xx$FitBMD,which="coef")[[1]]["BMD.sexM",])*xx$Dosescale
    }
  }

### Set up the dataframe to compute residuals, etc.

  seldata$Dose.scaled <- seldata$dose/xx$Dosescale
  seldata$set <- factor(paste(seldata$mrid,seldata$time,seldata$duplicate,sep=":"))
  seldata2  <- seldata
  seldata2$Dose.scaled[] <- 0

### Plots, etc. for Fitm
```

```
  if (inherits(xx$Fitm,"nlme")) {
    lvl <- switch(length(xx$Random[[3]][[3]]),1,2,2)
    preddata <- myPredict(xx$Fitm,newdata=seldata,
                          level=lvl)
    contdata <- myPredict(xx$Fitm,newdata=seldata2,
                          level=lvl)
  } else {
    preddata <- predict(xx$Fitm,newdata=seldata)
    contdata <- predict(xx$Fitm,newdata=seldata2)
  }
  Inh <- 1 - preddata/contdata
  resids <- sqrt(seldata$n)*(seldata$chei/xx$Respscale -
preddata)/(xx$Fitm$sigma*preddata^0.5)
  par(mfrow=c(2,2))
  plot(resids ~ Inh,ylab="Scaled Residuals",
       xlab="Predicted Fraction Inhibition",main=chem)
  abline(h=0,lty=2)
  plot(resids ~ I(preddata*xx$Respscale),ylab="Scaled Residuals",
       xlab="Predicted AChE Activity",main=chem)
  abline(h=0,lty=2)
  PP <- (((seldata$n-1)*(seldata$sd/xx$Respscale)^2)/
          (xx$Fitm$sigma^2*preddata)-(seldata$n-1))/sqrt(2*seldata$n-2)
  plot(PP~Inh,ylab="Variance Residual",
       xlab="Predicted Fraction Inhibition",main=chem)
  abline(h=0,lty=2)
  plot(PP~I(preddata*xx$Respscale),ylab="Variance Residual",
       xlab="Predicted AChE Activity",main=chem)
  abline(h=0,lty=2)
### for each unique level in s.U, plot response versus dose, and predicted
### response versus dose.  Plot both sexes on the same plot (red for females,
### blue for males).  Each unit gets its own plot.
  par(mfrow=c(1,1))
  for (Un in levels(seldata$Unit)) {
    yy <- seldata$chei[seldata$Unit == Un]
    x <- seldata$dose[seldata$Unit == Un]
    xpred <- seq(0,max(x),length=25)
    plot(x,yy,xlab="Dose (mg/kg/day)",ylab=paste("AChE (",Un,")",sep=""),
         type="n",main=paste(chem,Un,sep=" : "),ylim=range(0,yy))
    for (sex in c("F","M")) {
      yy <- seldata$chei[seldata$Unit == Un & seldata$sex == sex]
      if (length(yy) == 0) next
      x <- seldata$dose[seldata$Unit == Un & seldata$sex == sex]
      points(x,yy,col=c(F="red",M="blue")[sex])
      newdata <- data.frame(s.U = factor(rep(paste(sex,Un,sep="."),
                              length(xpred)),levels=levels(seldata$s.U)),
                            Dose.scaled = xpred/xx$Dosescale,
                            sex = factor(rep(sex,length(xpred)),
                              levels=c("F","M")))
      if (inherits(xx$Fitm,"nlme"))
        yypred <- myPredict(xx$Fitm,newdata=newdata,level=0)*xx$Respscale
      else
        yypred <- predict(xx$Fitm,newdata=newdata)*xx$Respscale
      lines(spline(xpred,yypred),col=c(F="red",M="blue")[sex])
      ## If there are multiple mrids, plot their curves as dotted lines
      if (inherits(xx$Fitm, "nlme")) {
        tmp <- seldata[seldata$Unit == Un & seldata$sex == sex,]
        if (length(unique(tmp$mrid)) > 1) {
          for (mrid in sort(unique(as.character(tmp$mrid)))) {
            newdata <- data.frame(s.U = factor(rep(paste(sex,Un,sep="."),
                                    length(xpred)),levels=levels(seldata$s.U)),
                                  Dose.scaled = xpred/xx$Dosescale,
                                  sex = factor(rep(sex,length(xpred)),
                                    levels=c("F","M")),
                                  mrid = factor(rep(mrid, length(xpred))))
            yypred <- myPredict(xx$Fitm,newdata=newdata,level=1)*xx$Respscale
            lines(spline(xpred,yypred),col=c(F="red",M="blue")[sex],lty=2)
          }
        }
      }
    }
  }

}
dev.off()
### Calculate the two kinds of relative potencies, and their 95% confidence
### limits
indexchem <- "METHAMIDOPHOS"
## Potency based on 'm'
## Females
PotencyFrame1[,"RPm.F"] <- exp(PotencyFrame1[,"lm.F"] -
                               PotencyFrame1[indexchem,"lm.F"])
lRPm.F.se <- sqrt(PotencyFrame1[,"lm.F.se"]^2 +
                  PotencyFrame1[indexchem,"lm.F.se"]^2)
PotencyFrame1[,"RPm.F.lcl"] <- exp(PotencyFrame1[,"lm.F"] -
                               PotencyFrame1[indexchem,"lm.F"] -
                                   1.96*lRPm.F.se)
PotencyFrame1[,"RPm.F.ucl"] <- exp(PotencyFrame1[,"lm.F"] -
                               PotencyFrame1[indexchem,"lm.F"] +
                                   1.96*lRPm.F.se)
PotencyFrame1[indexchem,c("RPm.F.lcl","RPm.F.ucl")] <- c(NA,NA)
## Males
PotencyFrame1[,"RPm.M"] <- exp(PotencyFrame1[,"lm.M"] -
                               PotencyFrame1[indexchem,"lm.M"])
lRPm.M.se <- sqrt(PotencyFrame1[,"lm.M.se"]^2 +
                  PotencyFrame1[indexchem,"lm.M.se"]^2)
PotencyFrame1[,"RPm.M.lcl"] <- exp(PotencyFrame1[,"lm.M"] -
                               PotencyFrame1[indexchem,"lm.M"] -
                                   1.96*lRPm.M.se)
PotencyFrame1[,"RPm.M.ucl"] <- exp(PotencyFrame1[,"lm.M"] -
                               PotencyFrame1[indexchem,"lm.M"] +
                                   1.96*lRPm.M.se)
PotencyFrame1[indexchem,c("RPm.M.lcl","RPm.M.ucl")] <- c(NA,NA)

## Potency based on 'BMD'
## Females
PotencyFrame1[,"RPBMD.F"] <- exp(PotencyFrame1[indexchem,"lBMD.F"] -
                               PotencyFrame1[,"lBMD.F"])
lRPBMD.F.se <- sqrt(PotencyFrame1[,"lBMD.F.se"]^2 +
                  PotencyFrame1[indexchem,"lBMD.F.se"]^2)
PotencyFrame1[,"RPBMD.F.lcl"] <- exp(PotencyFrame1[indexchem,"lBMD.F"] -
                               PotencyFrame1[,"lBMD.F"] -
                                   1.96*lRPBMD.F.se)
PotencyFrame1[,"RPBMD.F.ucl"] <- exp(PotencyFrame1[indexchem,"lBMD.F"] -
                               PotencyFrame1[,"lBMD.F"] +
                                   1.96*lRPBMD.F.se)
PotencyFrame1[indexchem,c("RPBMD.F.lcl","RPBMD.F.ucl")] <- c(NA,NA)
## Males
PotencyFrame1[,"RPBMD.M"] <- exp(PotencyFrame1[indexchem,"lBMD.M"] -
                               PotencyFrame1[,"lBMD.M"])
lRPBMD.M.se <- sqrt(PotencyFrame1[,"lBMD.M.se"]^2 +
                  PotencyFrame1[indexchem,"lBMD.M.se"]^2)
PotencyFrame1[,"RPBMD.M.lcl"] <- exp(PotencyFrame1[indexchem,"lBMD.M"] -
                               PotencyFrame1[,"lBMD.M"] -
                                   1.96*lRPBMD.M.se)
PotencyFrame1[,"RPBMD.M.ucl"] <- exp(PotencyFrame1[indexchem,"lBMD.M"] -
                               PotencyFrame1[,"lBMD.M"] +
                                   1.96*lRPBMD.M.se)
PotencyFrame1[indexchem,c("RPBMD.M.lcl","RPBMD.M.ucl")] <- c(NA,NA)

save(PotencyFrame1,file="PotencyFrame1.rda")
```

viii. pkgridsearch.R

This is analogous to BrainProfLik.R, and is invoked similarly. It computes the profile likelihoods for S and D in the expanded model.

```
### pkgridseach
###
### Calculate profile likelihood values for S vs. D for each chemical,
### to serve as starting values for fitting them.

cmdargs <- commandArgs()
if (length(grep("SimStart",cmdargs))==0 || length(grep("SimStop",cmdargs))==0)
  stop("Need arguments SimStart AND SimStop")
SimStart <- as.numeric(strsplit(cmdargs[grep("SimStart",cmdargs)],"=")[[1]][2])
SimStop <- as.numeric(strsplit(cmdargs[grep("SimStop",cmdargs)],"=")[[1]][2])

if (!is.finite(SimStart) || !is.finite(SimStop) || SimStart > SimStop)
  stop("SimStart or SimStop not correctly specified")

Chemicals <- sort(levels(BRAINdata$chemical))

for (chem in Chemicals[SimStart:SimStop]) {

  cat(paste("-------\n",chem,"\n\n"))

#### Get the data from the run in BRAIN-Models-2-save
  load(file.path("BRAIN-Models-2-save",chem))

  seldata <- xx$Data

  if ("fixed" %in% names(xx$Fitm$call$model[[3]])) {
    EstB <- FALSE
    fitNoB <- xx$Fitm
  } else {
    EstB <- TRUE
    fitWB <- xx$Fitm
  }

### loop through seldata (by block), using PhonyDF to expand it to
### synthetic individual data.


  tmp <- by(seldata,list(seldata$block),function(x) {
    PhonyDF(x$dose,x$n,x$chei,x$sd,"Dose","AChE",
            chem=rep(as.character(x$chunit[1]),nrow(x)),
            ChemName="Unit")
  })

  for (i in seq(along=tmp)) {
    flds <- unlist(strsplit(names(tmp[i]),"-"))
    N <- nrow(tmp[[i]])
    tmp[[i]]$mrid <- rep(flds[2],N)
    tmp[[i]]$sex <- rep(flds[4],N)
    tmp[[i]]$set <- rep(paste(flds[2],flds[5],flds[6],sep=":"),N)
  }

  Pseldata <- do.call("rbind",unclass(tmp))
  row.names(Pseldata) <- seq(along=Pseldata[[1]])
  Pseldata$set <- factor(Pseldata$set)
  Respscale <- max(seldata$chei)
  Dosescale <- max(seldata$dose)

  Pseldata$AChE.scaled <- Pseldata$AChE/Respscale
  Pseldata$Dose.scaled <- Pseldata$Dose/Dosescale
  Pseldata$s.U <- factor(interaction(Pseldata$sex, Pseldata$Unit, drop=TRUE))
```

```
#### Get starting values for A ~ sex:Units - 1
  controldata <- Pseldata[Pseldata$Dose == 0,]
  controldata$lAChE.scaled <- log(controldata$AChE.scaled)
  Astart.lme <- try(lme(lAChE.scaled ~ s.U - 1, data=controldata,
                      random=~1|set))
  if (inherits(Astart.lme,"try-error")) {

    Astart.lm <- lm(lAChE.scaled ~ s.U - 1, data=controldata)
    Astart.lme <- list(coefficients=list(fixed=coef(Astart.lm)))
  }
#### Get starting values for B
  Bs.F <- Bstart2[chem,"B.F"]
  Bs.F <- as.vector(log(Bs.F/(1 - Bs.F)))
  Bs.M <- Bstart2[chem,"B.M"]
  Bs.M <- as.vector(log(Bs.M/(1 - Bs.M)))

### values for A and m
### set up the random component:  These are the possibilities:
###
### one mrid, one unit, multiple sets:
###    RnDoM <- list(A+m ~ 1| set)
### multiple mrids, one mrid/unit (#mrids == #units), multiple sets:
###    RnDoM <- list(A ~ 1 | set, m ~ 1 | mrid/set
### multiple mrids, one mrid/unit (#mrids == #units), one set/mrid:
###    RnDoM <- list(m ~ 1 | mrid)
### multiple mrids, multiple mrids/unit (#mrids > #units), multiple sets:
###    RnDoM <- list(A+m ~ 1 | mrid/set)
### multiple mrids, multiple mrids/unit ($mrids > #units), one set/mrid:
###    RnDoM <- list(A+m ~ 1 | mrid)

  if (length(levels(Pseldata$mrid)) == 1 &&
      length(levels(Pseldata$set)) == 1) {
    RnDoM1 <- NULL
    RModel1 <- NULL
    RnDoM <- NULL
    RModel <- NULL
  }
  if (length(levels(Pseldata$mrid)) == 1 && length(levels(Pseldata$set)) > 1) {
    RnDoM1 <- list(set=pdDiag(form=A+m ~ 1))
    RModel1 <- A+m~1 | set
    RnDoM <- list(set=pdDiag(form=A+B+m ~ 1))
    RModel <- A+B+m~1 | set
  }
  if (length(levels(Pseldata$mrid)) > 1) {
    if (length(levels(Pseldata$mrid)) == length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) > length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=m ~ 1),
                     set=pdDiag(form=m~1))
      RModel1 <- m~1|mrid/set
      RnDoM <- list(mrid=pdDiag(form=B+m ~ 1),
                    set=pdDiag(form=B+m~1))
      RModel <- B+m~1|mrid/set
    }
    if (length(levels(Pseldata$mrid)) == length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) == length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=m ~ 1))
      RModel1 <- m~1|mrid
      RnDoM <- list(mrid=pdDiag(form=B+m ~ 1))
      RModel <- B+m~1|mrid
    }
    if (length(levels(Pseldata$mrid)) > length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) > length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=A+m~1),
                     set=pdDiag(form=A+m~1))
      RModel1 <- A+m~1|mrid/set
      RnDoM <- list(mrid=pdDiag(form=A+B+m~1),
                    set=pdDiag(form=A+B+m~1))
      RModel <- A+B+m~1|mrid/set
    }
    if (length(levels(Pseldata$mrid)) > length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) == length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=A+m ~ 1))
      RModel1 <- A+m~1|mrid
      RnDoM <- list(mrid=pdDiag(form=A+B+m ~ 1))
      RModel <- A+B+m~1|mrid
    }
  }
### Compute the grid of values.  Let S range over -8 to -1, 7 values,
### and D range over -8 to -1, 7 values.

  SDgrid <- expand.grid(S=seq(0.0001,0.4, length=7),
                        D=seq(0.0001,0.4, length=7))
  SDgrid$LL <- numeric(nrow(SDgrid))
  SDgrid$LL[] <- as.numeric(NA)
  for (i in 1:nrow(SDgrid)) {
### Next, the pk extension to CexpB: CpkexpB
    if (EstB) {
      ## Get start values from fitWB
      start <- if (inherits(fitWB, "nlme")) {
        fitWB$coefficients$fixed
      } else {
        fitWB$coefficients
      }
      ## set up the model
      Model <- eval(substitute(AChE.scaled ~
                               CpkexpBS(Dose.scaled,A,B,m,sex,
                                        fixed=
                                        list(D=c(F=xxxx,
                                              M=xxxx),
                                             S=c(F=yyyy,M=yyyy))),
                               list(xxxx=SDgrid$D[i],yyyy=SDgrid$S[i])))
      ## estimate it
      if (!is.null(RnDoM)) {
        fitpk <-
          try(eval(substitute(nlme(Model,data=Pseldata,
                                   fixed=list(A ~ s.U - 1, B ~ sex - 1,
                                     m ~ sex - 1),
                                   random=xxxx,
                                   start=zzzz,
                                   weights=varPower(fixed=0.5),
                                   method="ML",
                                   verbose=TRUE),
                              list(Model=Model,xxxx=RnDoM,
                                   zzzz=Tstart))))
      } else {
        fitpk <-
          try(eval(substitute(gnls(Model,data=Pseldata,
                                   params=list(A ~ s.U - 1, B ~ sex - 1,
                                     m ~ sex - 1),
                                   start=zzzz,
                                   weights=varPower(fixed=0.5),
                                   verbose=TRUE),
                              list(Model=Model,
                                   zzzz=Tstart))))
      }
    } else {
      ## Get start values from fitNoB
      start <- if (inherits(fitNoB, "nlme")) fitNoB$coefficients$fixed else
      fitNoB$coefficients
      ## set up the model
      Model <- eval(substitute(AChE.scaled ~ CpkexpBS(Dose.scaled,A,m,sex,
                                                      fixed=list(B=c(F=xxxx,
                                                                   M=yyyy),
                                                        S=c(F=zzzz, M=zzzz),
                                                        D=c(F=wwww, M=wwww))),
                               list(xxxx=Bs.F,yyyy=Bs.M,zzzz=SDgrid$S[i],
                                   wwww=SDgrid$D[i])))
      ## estimate it
      if (!is.null(RnDoM1)) {
        fitpk <-
          try(eval(substitute(nlme(Model,data=Pseldata,
                                   fixed=list(A ~ s.U - 1, m ~ sex - 1),
                                   random=xxxx,
                                   start=zzzz,
                                   weights=varPower(fixed=0.5),
                                   method="ML",verbose=TRUE),
                              list(Model=Model,xxxx=RnDoM1,
                                   zzzz=start))))
      } else {
        fitpk <-
          try(eval(substitute(gnls(Model, data=Pseldata,
                                   params=list(A ~ s.U - 1, m ~ sex - 1),
                                   start=zzzz,
                                   weights=varPower(fixed=0.5),
                                   verbose=TRUE),
                              list(Model=Model,
                                   zzzz=start))))
      }
    }
    if (!inherits(fitpk, "try-error")) SDgrid$LL[i] <- logLik(fitpk)
  }

  xx <- list(Chemical=chem, EstB=EstB,
             LLgrid=SDgrid,
             Respscale=Respscale,Dosescale=Dosescale, Random=RModel,
             Data=seldata)
  save(xx,file=file.path("BRAIN-pkgrids",chem))
}
```

ix. plotSDProfiles.R

This is similar to plotProfiles.R and makes a pdf file of the figures.

```
### plotProfiles.R
require(akima)
pdf(file="Profiles-4-SD.pdf")
dirname <- "BRAIN-pkgrids"
for (chem in Chemicals){
  if (!file.exists(file.path(dirname,chem))) next
  load(file.path(dirname,chem))
  LLgrid <- xx$LLgrid
  LLgrid <- na.omit(LLgrid)
  if (length(LLgrid$LL) == 0) {
    plot(c(0,1),c(0,1),type="n",axes=FALSE, xlab="", ylab="")
    text(0.5,0.5,paste(chem,"no fits"),adj=0.5)
  } else {
    out <- try(interp(LLgrid$S,LLgrid$D,LLgrid$LL,
                  seq(min(LLgrid$S), max(LLgrid$S),length=40),
                  seq(min(LLgrid$D), max(LLgrid$D),length=40)))
    if (!inherits(out, "try-error")) {
      res <- try(image(out,xlab="S",ylab="D",main=chem))
      if (inherits(res, "try-error")) {
        plot(c(0,1),c(0,1),type="n",axes=FALSE, xlab="", ylab="")
        text(0.5,0.5,paste(chem,"not enough fits"),adj=0.5)
      } else {
        points(LLgrid$S,LLgrid$D,pch=3,cex=0.5)
      }
    } else {
    plot(c(0,1),c(0,1),type="n",axes=FALSE, xlab="", ylab="")
    text(0.5,0.5,paste(chem,"not enough fits"),adj=0.5)
    }
  }
}
dev.off()
```

x. plotSDProfiles2.R

This is an interactive version of the previous script, and needs to be sourced at the R command line, like plotProfiles2.R.

```
### plotSDProfiles2.R
require(akima)
SDstart <- data.frame(S=vector("numeric",length(Chemicals)),
                      D=vector("numeric",length(Chemicals)),
                      row.names=Chemicals)
dirname <- "BRAIN-pkgrids"
for (chem in Chemicals){
  if (!file.exists(file.path(dirname,chem))) next
  load(file.path(dirname,chem))
  LLgrid <- xx$LLgrid
  LLgrid <- na.omit(LLgrid)
  if (length(LLgrid$LL) == 0) {
    SDstart[chem,c("S","D")] <- c(NA,NA)
    next
  }
  out <- interp(LLgrid$S,LLgrid$D,LLgrid$LL,
                seq(min(LLgrid$S), max(LLgrid$S),length=40),
                seq(min(LLgrid$D), max(LLgrid$D),length=40))
  res <- try(image(out,xlab="S",ylab="D",main=chem))
  if (inherits(res, "try-error")) {
    SDstart[chem,] <- NA
    next
  }
  points(LLgrid$S,LLgrid$D,pch=3,cex=0.5)
  tmp <- locator(1)
  if (length(tmp) > 0) {
    SDstart[chem,"S"] <- tmp$x
    SDstart[chem,"D"] <- tmp$y
  } else SDstart[chem,c("S","D")] <- c(NA,NA)
}
save(SDstart,file="SDstart.rda")
```

xi. BRAINpkfits.R

This fits the expanded model to the data, and depends on the previous scripts to have already been run. As for the script that fits the basic model, this is set up for specifying ‘SimStart’ and ‘SimStop’ on the command line.

```
### BRAINpkfits
###
### Start with the exp model and estimate D (fixing S to something reasonable,
### determined by examining profile likelihoods for a range of S and D).

cmdargs <- commandArgs()
if (length(grep("SimStart",cmdargs))==0 || length(grep("SimStop",cmdargs))==0)
  stop("Need arguments SimStart AND SimStop")
SimStart <- as.numeric(strsplit(cmdargs[grep("SimStart",cmdargs)],"=")[[1]][2])
SimStop <- as.numeric(strsplit(cmdargs[grep("SimStop",cmdargs)],"=")[[1]][2])

if (!is.finite(SimStart) || !is.finite(SimStop) || SimStart > SimStop)
  stop("SimStart or SimStop not correctly specified")

Chemicals <- sort(levels(BRAINdata$chemical))
load("SDstart.rda")

for (chem in Chemicals[SimStart:SimStop]) {
  if (is.na(SDstart[chem,"S"])) next
  cat(paste("-------\n",chem,"\n\n"))

#### Get the data from the run in BRAIN-Models-2-save
  load(file.path("BRAIN-Models-2-save",chem))

  seldata <- xx$Data

  if ("fixed" %in% names(xx$Fitm$call$model[[3]])) {
    EstB <- FALSE
    fitNoB <- xx$Fitm
  } else {
    EstB <- TRUE
    fitWB <- xx$Fitm
  }

### loop through seldata (by block), using PhonyDF to expand it to
### synthetic individual data.


  tmp <- by(seldata,list(seldata$block),function(x) {
    PhonyDF(x$dose,x$n,x$chei,x$sd,"Dose","AChE",
            chem=rep(as.character(x$chunit[1]),nrow(x)),
            ChemName="Unit")
  })

  for (i in seq(along=tmp)) {
    flds <- unlist(strsplit(names(tmp[i]),"-"))
    N <- nrow(tmp[[i]])
    tmp[[i]]$mrid <- rep(flds[2],N)
    tmp[[i]]$sex <- rep(flds[4],N)
    tmp[[i]]$set <- rep(paste(flds[2],flds[5],flds[6],sep=":"),N)
  }

  Pseldata <- do.call("rbind",unclass(tmp))
  row.names(Pseldata) <- seq(along=Pseldata[[1]])
  Pseldata$set <- factor(Pseldata$set)
  Respscale <- xx$Respscale
  Dosescale <- xx$Dosescale

  Pseldata$AChE.scaled <- Pseldata$AChE/Respscale
  Pseldata$Dose.scaled <- Pseldata$Dose/Dosescale
  Pseldata$s.U <- factor(interaction(Pseldata$sex, Pseldata$Unit, drop=TRUE))

#### Get starting values for B
  Bs.F <- Bstart2[chem,"B.F"]
  Bs.F <- as.vector(log(Bs.F/(1 - Bs.F)))
  Bs.M <- Bstart2[chem,"B.M"]
  Bs.M <- as.vector(log(Bs.M/(1 - Bs.M)))

### values for A and m
### set up the random component:  These are the possibilities:
###
### one mrid, one unit, multiple sets:
###    RnDoM <- list(A+m ~ 1| set)
### multiple mrids, one mrid/unit (#mrids == #units), multiple sets:
###    RnDoM <- list(A ~ 1 | set, m ~ 1 | mrid/set
### multiple mrids, one mrid/unit (#mrids == #units), one set/mrid:
###    RnDoM <- list(m ~ 1 | mrid)
### multiple mrids, multiple mrids/unit (#mrids > #units), multiple sets:
###    RnDoM <- list(A+m ~ 1 | mrid/set)
### multiple mrids, multiple mrids/unit ($mrids > #units), one set/mrid:
###    RnDoM <- list(A+m ~ 1 | mrid)

  if (length(levels(Pseldata$mrid)) == 1 &&
      length(levels(Pseldata$set)) == 1) {
    RnDoM1 <- NULL
    RModel1 <- NULL
    RnDoM <- NULL
    RModel <- NULL
  }
  if (length(levels(Pseldata$mrid)) == 1 && length(levels(Pseldata$set)) > 1) {
    RnDoM1 <- list(set=pdDiag(form=A+m ~ 1))
    RModel1 <- A+m~1 | set
    RnDoM <- list(set=pdDiag(form=A+B+m ~ 1))
    RModel <- A+B+m~1 | set
  }
  if (length(levels(Pseldata$mrid)) > 1) {
    if (length(levels(Pseldata$mrid)) == length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) > length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=m ~ 1),
                     set=pdDiag(form=m~1))
      RModel1 <- m~1|mrid/set
      RnDoM <- list(mrid=pdDiag(form=B+m ~ 1),
                    set=pdDiag(form=B+m~1))
      RModel <- B+m~1|mrid/set
    }
    if (length(levels(Pseldata$mrid)) == length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) == length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=m ~ 1))
      RModel1 <- m~1|mrid
      RnDoM <- list(mrid=pdDiag(form=B+m ~ 1))
      RModel <- B+m~1|mrid
    }
    if (length(levels(Pseldata$mrid)) > length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) > length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=A+m~1),
                     set=pdDiag(form=A+m~1))
      RModel1 <- A+m~1|mrid/set
      RnDoM <- list(mrid=pdDiag(form=A+B+m~1),
                    set=pdDiag(form=A+B+m~1))
      RModel <- A+B+m~1|mrid/set
    }
    if (length(levels(Pseldata$mrid)) > length(levels(Pseldata$Unit)) &&
        length(levels(Pseldata$set)) == length(levels(Pseldata$mrid))) {
      RnDoM1 <- list(mrid=pdDiag(form=A+m ~ 1))
      RModel1 <- A+m~1|mrid
      RnDoM <- list(mrid=pdDiag(form=A+B+m ~ 1))
      RModel <- A+B+m~1|mrid
    }
  }
  ## Add 'D' to start, and get the value of S to use.
  fixS <- if (SDstart[chem,"S"] < 0.0001) -15 else
  log(SDstart[chem,"S"])
  if (EstB) {
    ## Get start values from fitWB
    start <- if (inherits(fitWB, "nlme")) {
      fitWB$coefficients$fixed
    } else {
      fitWB$coefficients
    }
    start2 <- start
    start["D"] <- log(SDstart[chem,"D"])
    ## set up the model
    Model <- eval(substitute(AChE.scaled ~
                             CpkexpBS(Dose.scaled,A,B,m,D,sex,
                                      fixed=list(S=c(F=yyyy,M=yyyy))),
                             list(yyyy=fixS)))
    Model2 <- eval(substitute(AChE.scaled ~
                             CpkexpBS(Dose.scaled,A,B,m,sex,
                                      fixed=list(S=c(F=yyyy,M=yyyy),
                                        D=c(F=-15,M=-15))),
                             list(yyyy=fixS)))
    ## estimate it
    if (!is.null(RnDoM)) {
      fitpk <-
        try(eval(substitute(nlme(Model,data=Pseldata,
                                 fixed=list(A ~ s.U - 1, B ~ sex - 1,
                                   m ~ sex - 1, D ~ 1),
                                 random=xxxx,
                                 start=zzzz,
                                 weights=varPower(fixed=0.5),
                                 method="ML",
                                 verbose=TRUE),
                            list(Model=Model,xxxx=RnDoM,
                                 zzzz=start))))
      fitpk0 <-
        try(eval(substitute(nlme(Model,data=Pseldata,
                                 fixed=list(A ~ s.U - 1, B ~ sex - 1,
                                   m ~ sex - 1),
                                 random=xxxx,
                                 start=zzzz,
                                 weights=varPower(fixed=0.5),
                                 method="ML",
                                 verbose=TRUE),
                            list(Model=Model2,xxxx=RnDoM,
                                 zzzz=start2))))
    } else {
      fitpk <-
        try(eval(substitute(gnls(Model,data=Pseldata,
                                 params=list(A ~ s.U - 1, B ~ sex - 1,
                                   m ~ sex - 1, D ~ 1),
                                 start=zzzz,
                                 weights=varPower(fixed=0.5),
                                 verbose=TRUE),
                            list(Model=Model,
                                 zzzz=start))))
      fitpk0 <-
        try(eval(substitute(gnls(Model,data=Pseldata,
                                 params=list(A ~ s.U - 1, B ~ sex - 1,
                                   m ~ sex - 1),
                                 start=zzzz,
                                 weights=varPower(fixed=0.5),
                                 verbose=TRUE),
                            list(Model=Model2,
                                 zzzz=start2))))
    }
```

```
} else {
  ## Get start values from fitNoB
  start <- if (inherits(fitNoB, "nlme")) fitNoB$coefficients$fixed else
  fitNoB$coefficients
  start2 <- start
  start["D"] <- log(SDstart[chem,"D"])
  ## set up the model
  Model <- eval(substitute(AChE.scaled ~ CpkexpBS(Dose.scaled,A,m,D,sex,
                                                fixed=list(B=c(F=xxxx,
                                                             M=yyyy),
                                                  S=c(F=zzzz, M=zzzz))),
                          list(xxxx=Bs.F,yyyy=Bs.M,zzzz=fixS)))
  Model2 <- eval(substitute(AChE.scaled ~ CpkexpBS(Dose.scaled,A,m,sex,
                                                fixed=list(B=c(F=xxxx,
                                                             M=yyyy),
                                                  S=c(F=zzzz, M=zzzz),
                                                  D=c(F=-15,M=-15))),
                          list(xxxx=Bs.F,yyyy=Bs.M,zzzz=fixS)))
  ## estimate it
  if (!is.null(RnDoM1)) {
    fitpk <-
      try(eval(substitute(nlme(Model,data=Pseldata,
                               fixed=list(A ~ s.U - 1, m ~ sex - 1, D ~ 1),
                               random=xxxx,
                               start=zzzz,
                               weights=varPower(fixed=0.5),
                               method="ML",verbose=TRUE),
                          list(Model=Model,xxxx=RnDoM1,
                               zzzz=start))))
    fitpk0 <-
      try(eval(substitute(nlme(Model,data=Pseldata,
                               fixed=list(A ~ s.U - 1, m ~ sex - 1),
                               random=xxxx,
                               start=zzzz,
                               weights=varPower(fixed=0.5),
                               method="ML",verbose=TRUE),
                          list(Model=Model2,xxxx=RnDoM1,
                               zzzz=start2))))
  } else {
    fitpk <-
      try(eval(substitute(gnls(Model, data=Pseldata,
                               params=list(A ~ s.U - 1, m ~ sex - 1, D ~ 1),
                               start=zzzz,
                               weights=varPower(fixed=0.5),
                               verbose=TRUE),
                          list(Model=Model,
                               zzzz=start))))
    fitpk0 <-
      try(eval(substitute(gnls(Model, data=Pseldata,
                               params=list(A ~ s.U - 1, m ~ sex - 1),
                               start=zzzz,
                               weights=varPower(fixed=0.5),
                               verbose=TRUE),
                          list(Model=Model2,
                               zzzz=start2))))
  }
}
### If that didn't work, try a range of initial values for D, centered
### on our value, and ranging from 75% to 150%  of our initial value.
if (inherits(fitpk, "try-error")) {
  trycount <- 0
  trymax <- 20
  Dstart <- log(seq(0.7,0.1,length=trymax+1))
  LL <- numeric(length(Dstart))
  LL[] <- NA
  repeat {
    trycount <- trycount + 1
    if (EstB) {
```

```
    ## Get start values from fitWB
    start <- if (inherits(fitWB, "nlme")) {
      fitWB$coefficients$fixed
    } else {
      fitWB$coefficients
    }
    start["D"] <- Dstart[trycount]
    ## set up the model
    Model <- eval(substitute(AChE.scaled ~
                             CpkexpBS(Dose.scaled,A,B,m,D,sex,
                                      fixed=list(S=c(F=yyyy,M=yyyy))),
                             list(yyyy=fixS)))
    Model2 <- eval(substitute(AChE.scaled ~
                              CpkexpBS(Dose.scaled,A,B,m,sex,
                                       fixed=list(S=c(F=yyyy,M=yyyy),
                                         D=c(F=-15,M=-15))),
                              list(yyyy=fixS)))
    ## estimate it
    if (!is.null(RnDoM)) {
      fitpk <-
        try(eval(substitute(nlme(Model,data=Pseldata,
                                 fixed=list(A ~ s.U - 1, B ~ sex - 1,
                                   m ~ sex - 1, D ~ 1),
                                 random=xxxx,
                                 start=zzzz,
                                 weights=varPower(fixed=0.5),
                                 method="ML",
                                 verbose=TRUE),
                            list(Model=Model,xxxx=RnDoM,
                                 zzzz=start))))
    } else {
      fitpk <-
        try(eval(substitute(gnls(Model,data=Pseldata,
                                 params=list(A ~ s.U - 1, B ~ sex - 1,
                                   m ~ sex - 1, D ~ 1),
                                 start=zzzz,
                                 weights=varPower(fixed=0.5),
                                 verbose=TRUE),
                            list(Model=Model,zzzz=start))))
    }
  } else {
    ## Get start values from fitNoB
    start <- if (inherits(fitNoB, "nlme")) fitNoB$coefficients$fixed else
    fitNoB$coefficients
    start["D"] <- Dstart[trycount]
    ## set up the model
    Model <- eval(substitute(AChE.scaled ~ CpkexpBS(Dose.scaled,A,m,D,sex,
                                                    fixed=list(B=c(F=xxxx,
                                                                 M=yyyy),
                                                      S=c(F=zzzz,
                                                        M=zzzz))),
                             list(xxxx=Bs.F,yyyy=Bs.M,zzzz=fixS)))
    ## estimate it
    if (!is.null(RnDoM1)) {
      fitpk <-
        try(eval(substitute(nlme(Model,data=Pseldata,
                                 fixed=list(A ~ s.U - 1, m ~ sex - 1,
                                   D ~ 1),
                                 random=xxxx,
                                 start=zzzz,
                                 weights=varPower(fixed=0.5),
                                 method="ML",verbose=TRUE),
                            list(Model=Model,xxxx=RnDoM1,
                                 zzzz=start))))
    } else {
      fitpk <-
        try(eval(substitute(gnls(Model, data=Pseldata,
                                 params=list(A ~ s.U - 1, m ~ sex - 1,
                                  D ~ 1),
                                start=zzzz,
                                weights=varPower(fixed=0.5),
                                verbose=TRUE),
                           list(Model=Model,
                                zzzz=start))))
    }
  }
  if (!inherits(fitpk, "try-error")) {
    LL[trycount] <- logLik(fitpk)
  }
  if (trycount >= trymax) break
}
Dstart <- Dstart[!is.na(LL)]
LL <- LL[!is.na(LL)]
if (length(LL) > 0) {
  Dbest <- Dstart[which.max(LL)]
  if (EstB) {
    ## Get start values from fitWB
    start <- if (inherits(fitWB, "nlme")) {
      fitWB$coefficients$fixed
    } else {
      fitWB$coefficients
    }
    start["D"] <- Dbest
    ## set up the model
    Model <- eval(substitute(AChE.scaled ~
                         CpkexpBS(Dose.scaled,A,B,m,D,sex,
                                  fixed=list(S=c(F=yyyy,M=yyyy))),
                         list(yyyy=fixS)))
    Model2 <- eval(substitute(AChE.scaled ~
                          CpkexpBS(Dose.scaled,A,B,m,sex,
                                   fixed=list(S=c(F=yyyy,M=yyyy),
                                     D=c(F=-15,M=-15))),
                          list(yyyy=fixS)))
    ## estimate it
    if (!is.null(RnDoM)) {
      fitpk <-
        try(eval(substitute(nlme(Model,data=Pseldata,
                                fixed=list(A ~ s.U - 1, B ~ sex - 1,
                                  m ~ sex - 1, D ~ 1),
                                random=xxxx,
                                start=zzzz,
                                weights=varPower(fixed=0.5),
                                method="ML",
                                verbose=TRUE),
                           list(Model=Model,xxxx=RnDoM,
                                zzzz=start))))
    } else {
      fitpk <-
        try(eval(substitute(gnls(Model,data=Pseldata,
                                params=list(A ~ s.U - 1, B ~ sex - 1,
                                  m ~ sex - 1, D ~ 1),
                                start=zzzz,
                                weights=varPower(fixed=0.5),
                                verbose=TRUE),
                           list(Model=Model,zzzz=start))))
    }
  } else {
    ## Get start values from fitNoB
    start <- if (inherits(fitNoB, "nlme")) fitNoB$coefficients$fixed else
    fitNoB$coefficients
    start["D"] <- Dbest
    ## set up the model
    Model <- eval(substitute(AChE.scaled ~ CpkexpBS(Dose.scaled,A,m,D,sex,
                                               fixed=list(B=c(F=xxxx,
                                                            M=yyyy),
                                                 S=c(F=zzzz,
                                                         M=zzzz))),
                                 list(xxxx=Bs.F,yyyy=Bs.M,zzzz=fixS)))
        ## estimate it
        if (!is.null(RnDoM1)) {
          fitpk <-
            try(eval(substitute(nlme(Model,data=Pseldata,
                                     fixed=list(A ~ s.U - 1, m ~ sex - 1,
                                       D ~ 1),
                                     random=xxxx,
                                     start=zzzz,
                                     weights=varPower(fixed=0.5),
                                     method="ML",verbose=TRUE),
                                list(Model=Model,xxxx=RnDoM1,
                                     zzzz=start))))
        } else {
          fitpk <-
            try(eval(substitute(gnls(Model, data=Pseldata,
                                     params=list(A ~ s.U - 1, m ~ sex - 1,
                                       D ~ 1),
                                     start=zzzz,
                                     weights=varPower(fixed=0.5),
                                     verbose=TRUE),
                                list(Model=Model,
                                     zzzz=start))))
        }
      }
    }
  }

  xx <- list(Chemical=chem, EstB=EstB,
             Fitpk=fitpk,
             Fitpk0=fitpk0,
             Respscale=Respscale,Dosescale=Dosescale,
             Random=if (EstB) RModel else RModel1,
             Data=seldata)
  save(xx,file=file.path("BRAIN-pkfits-2",chem))
}
```

xii. plotpkresids2.R

Plots the residuals, etc. from the expanded model, and constructs a data frame, PotencyFrame2, that contains various important model parameters. This is largely superceeded by the matrices created in the final script in this section. Indeed, since plotpkresids2.R was written, the function for estimating BMDs for the expanded model changed, so the values produced here are incorrect.

```
### plotpkresids2.R

### Create a BRAIN only set from opdata

BRAINdata <- opdata[opdata$compartment == "BRAIN" &
                    opdata$duplicate %in% c("WHOLE","DUPLICATEWHOLE",
                                            "WHOLEDUPLICATE"),]

BRAINdata$chemical <- factor(BRAINdata$chemical)
BRAINdata$chunit <- factor(BRAINdata$chunit, exclude=NULL)
levels(BRAINdata$chunit)[grep("NA",levels(BRAINdata$chunit))] <- "Unk"
Chemicals <- sort(levels(BRAINdata$chemical))
load("PotencyFrame1.rda")

### Create a data set to hold the m's BMDS, potencies, etc.
nchem <- length(Chemicals)
PotencyFrame2 <- data.frame(B.est=as.logical(rep(NA,nchem)),
                            Random=as.logical(rep(NA,nchem)),
                            m.F=as.numeric(rep(NA,nchem)),
                            lm.F=as.numeric(rep(NA,nchem)),
                            lm.F.se=as.numeric(rep(NA,nchem)),
                            m.F.lcl=as.numeric(rep(NA,nchem)),
                            m.F.ucl=as.numeric(rep(NA,nchem)),
                            m.M=as.numeric(rep(NA,nchem)),
                            lm.M=as.numeric(rep(NA,nchem)),
                            lm.M.se=as.numeric(rep(NA,nchem)),
                            m.M.lcl=as.numeric(rep(NA,nchem)),
                            m.M.ucl=as.numeric(rep(NA,nchem)),
                            RPm.F=as.numeric(rep(NA,nchem)),
                            RPm.F.lcl=as.numeric(rep(NA,nchem)),
                            RPm.F.ucl=as.numeric(rep(NA,nchem)),
                            RPm.M=as.numeric(rep(NA,nchem)),
                            RPm.M.lcl=as.numeric(rep(NA,nchem)),
                            RPm.M.ucl=as.numeric(rep(NA,nchem)),
                            BMD.F=as.numeric(rep(NA,nchem)),
                            lBMD.F=as.numeric(rep(NA,nchem)),
                            lBMD.F.se=as.numeric(rep(NA,nchem)),
                            BMD.F.lcl=as.numeric(rep(NA,nchem)),
                            BMD.F.ucl=as.numeric(rep(NA,nchem)),
                            BMD.M=as.numeric(rep(NA,nchem)),
                            lBMD.M=as.numeric(rep(NA,nchem)),
                            lBMD.M.se=as.numeric(rep(NA,nchem)),
                            BMD.M.lcl=as.numeric(rep(NA,nchem)),
                            BMD.M.ucl=as.numeric(rep(NA,nchem)),
                            RPBMD.F=as.numeric(rep(NA,nchem)),
                            RPBMD.F.lcl=as.numeric(rep(NA,nchem)),
                            RPBMD.F.ucl=as.numeric(rep(NA,nchem)),
                            RPBMD.M=as.numeric(rep(NA,nchem)),
                            RPBMD.M.lcl=as.numeric(rep(NA,nchem)),
                            RPBMD.M.ucl=as.numeric(rep(NA,nchem)),
                            row.names=Chemicals)
PotencyFrame2$B.est <- as.logical(PotencyFrame2$B.est)
PotencyFrame2$Random <- as.logical(PotencyFrame2$Random)
```

```
pdf("BRAINpkresids2.pdf")

for (chem in Chemicals) {

  cat(paste("-------\n",chem,"\n\n"))
  if (!file.exists(file.path("BRAIN-pkfits-2",chem)) ||
      file.info(file.path("BRAIN-pkfits-2",chem))[1,"size"] < 4*1024) {
    cat("No Model Fit\n")
    next
  }
  sel <- BRAINdata$chemical == chem
  seldata <- BRAINdata[sel,]
  seldata$block <- factor(seldata$block)
  seldata$mrid <- factor(seldata$mrid)
  seldata$Unit <- factor(seldata$chunit)
  seldata$s.U <- factor(interaction(seldata$sex, seldata$Unit, drop=TRUE))

### load the model fit

  load(file.path("BRAIN-pkfits-2",chem))

### There is only Fitpk; if Fitpk didn't work, go to the next chemical.

  if (inherits(xx$Fitpk,"try-error")) next

### Fill the corresponding record for PotencyFrame2

  PotencyFrame2[chem,"B.est"] <- !("fixed" %in% names(xx$Fitpk$call$model[[3]]))
  PotencyFrame2[chem,"Random"] <- !is.null(xx$Random)
  tTable <- summary(xx$Fitpk)$tTable
  PotencyFrame2[chem,"lm.F"] <- tTable["m.sexF",1] - log(xx$Dosescale)
  PotencyFrame2[chem,"lm.M"] <- tTable["m.sexM",1] - log(xx$Dosescale)
  PotencyFrame2[chem,"lm.F.se"] <- tTable["m.sexF",2]
  PotencyFrame2[chem,"lm.M.se"] <- tTable["m.sexM",2]
  if (inherits(xx$Fitpk,"nlme")) {
    PotencyFrame2[chem,c("m.F.lcl","m.F","m.F.ucl")] <-
      exp(intervals(xx$Fitpk,which="fixed")[[1]]["m.sexF",])/xx$Dosescale
    PotencyFrame2[chem,c("m.M.lcl","m.M","m.M.ucl")] <-
      exp(intervals(xx$Fitpk,which="fixed")[[1]]["m.sexM",])/xx$Dosescale
  } else {
    PotencyFrame2[chem,c("m.F.lcl","m.F","m.F.ucl")] <-
      exp(intervals(xx$Fitpk,which="coef")[[1]]["m.sexF",])/xx$Dosescale
    PotencyFrame2[chem,c("m.M.lcl","m.M","m.M.ucl")] <-
      exp(intervals(xx$Fitpk,which="coef")[[1]]["m.sexM",])/xx$Dosescale
  }
  lBMDs <- pkexpSlBMD.se(xx,BMR=0.10)
  PotencyFrame2[chem,"lBMD.F"] <- lBMDs$lBMD["F"] + log(xx$Dosescale)
  PotencyFrame2[chem,"lBMD.M"] <- lBMDs$lBMD["M"] + log(xx$Dosescale)
##  PotencyFrame2[chem,"lBMD.F.se"] <- lBMDs$lBMD.se["F"]
##  PotencyFrame2[chem,"lBMD.M.se"] <- lBMDs$lBMD.se["M"]
  PotencyFrame2[chem,c("BMD.F.lcl","BMD.F","BMD.F.ucl")] <-
    c(NA,exp(PotencyFrame2[chem,"lBMD.F"]),NA)
  PotencyFrame2[chem,c("BMD.M.lcl","BMD.M","BMD.M.ucl")] <-
    c(NA,exp(PotencyFrame2[chem,"lBMD.M"]),NA)

### Set up the dataframe to compute residuals, etc.

  seldata$Dose.scaled <- seldata$dose/xx$Dosescale
  seldata$set <- factor(paste(seldata$mrid,seldata$time,seldata$duplicate,sep=":"))
  seldata2  <- seldata
  seldata2$Dose.scaled[] <- 0

### Plots, etc. for Fitpk
  if (inherits(xx$Fitpk,"nlme")) {
    lvl <- switch(length(xx$Random[[3]][[3]]),1,2,2)
## Sigh...There is something wrong with predict.nlme
#      preddata <- predict(xx$Fitpk,newdata=seldata,
#                          level=length(xx$Random[[3]][[3]])-1)
#      contdata <- predict(xx$Fitpk,newdata=seldata2,
#                          level=length(xx$Random[[3]][[3]])-1)
     preddata <- mypkPredict(xx$Fitpk,newdata=seldata,
                         level=lvl)
     contdata <- mypkPredict(xx$Fitpk,newdata=seldata2,
                         level=lvl)

  } else {
    preddata <- predict(xx$Fitpk,newdata=seldata)
    contdata <- predict(xx$Fitpk,newdata=seldata2)
  }
  Inh <- 1 - preddata/contdata
  resids <- sqrt(seldata$n)*(seldata$chei/xx$Respscale -
                           preddata)/(xx$Fitpk$sigma*preddata^0.5)
  par(mfrow=c(2,2))
  plot(resids ~ Inh,ylab="Scaled Residuals",
       xlab="Predicted Fraction Inhibition",main=chem)
  abline(h=0,lty=2)
  plot(resids ~ I(preddata*xx$Respscale),ylab="Scaled Residuals",
       xlab="Predicted AChE Activity",main=chem)
  abline(h=0,lty=2)
  PP <- (((seldata$n-1)*(seldata$sd/xx$Respscale)^2)/
          (xx$Fitpk$sigma^2*preddata)-(seldata$n-1))/sqrt(2*seldata$n-2)
  plot(PP~Inh,ylab="Variance Residual",xlab="Predicted Fraction Inhibition",main=chem)
  abline(h=0,lty=2)
  plot(PP~I(preddata*xx$Respscale),ylab="Variance Residual",
        xlab="Predicted AChE Activity",main=chem)
  abline(h=0,lty=2)
### for each unique level in s.U, plot response versus dose, and predicted
### response versus dose.  Plot both sexes on the same plot (red for females,
### blue for males).  Each unit gets its own plot.
  par(mfrow=c(1,1))
  for (Un in levels(seldata$Unit)) {
    yy <- seldata$chei[seldata$Unit == Un]
    x <- seldata$dose[seldata$Unit == Un]
    xpred <- seq(0,max(x),length=101)
    plot(x,yy,xlab="Dose (mg/kg/day)",ylab=paste("AChE (",Un,")",sep=""),
         type="n",main=paste(chem,Un,sep=" : "),ylim=range(0,yy))
    for (sex in c("F","M")) {
      yy <- seldata$chei[seldata$Unit == Un & seldata$sex == sex]
      if (length(yy) == 0) next
      x <- seldata$dose[seldata$Unit == Un & seldata$sex == sex]
      points(x,yy,col=c(F="red",M="blue")[sex])
      newdata <- data.frame(s.U = factor(rep(paste(sex,Un,sep="."),
                             length(xpred)),levels=levels(seldata$s.U)),
                           Dose.scaled = xpred/xx$Dosescale,
                           sex = factor(rep(sex,length(xpred)),
                             levels=c("F","M")))
      if (inherits(xx$Fitpk,"nlme"))
        yypred <- mypkPredict(xx$Fitpk,newdata=newdata,level=0)*xx$Respscale
      else
        yypred <- predict(xx$Fitpk,newdata=newdata)*xx$Respscale
      lines(xpred,yypred,col=c(F="red",M="blue")[sex])
      ## If there are multiple mrids, plot their curves as dotted lines
      if (inherits(xx$Fitpk, "nlme")) {
        tmp <- seldata[seldata$Unit == Un & seldata$sex == sex,]
        if (length(unique(tmp$mrid)) > 1) {
          for (mrid in sort(unique(as.character(tmp$mrid)))) {
            newdata <- data.frame(s.U = factor(rep(paste(sex,Un,sep="."),
                                   length(xpred)),levels=levels(seldata$s.U)),
                                 Dose.scaled = xpred/xx$Dosescale,
                                 sex = factor(rep(sex,length(xpred)),
                                   levels=c("F","M")),
                                 mrid = factor(rep(mrid, length(xpred))))
            yypred <- mypkPredict(xx$Fitpk,newdata=newdata,level=1)*xx$Respscale
            lines(xpred,yypred,col=c(F="red",M="blue")[sex],lty=2)
          }
        }
      }
    }
  }

}
dev.off()
### Calculate the two kinds of relative potencies, and their 95% confidence
### limits
indexchem <- "METHAMIDOPHOS"
### Use the values in PotencyFrame2 as denominators if indexchem has
### values there, else use the values in PotencyFrame1
if (!is.na(PotencyFrame2[indexchem,"lm.F"])) {
  indexvals <- unlist(PotencyFrame2[indexchem,])
} else {
  indexvals <- unlist(PotencyFrame1[indexchem,])
}
## Potency based on 'm'
## Females
PotencyFrame2[,"RPm.F"] <- exp(PotencyFrame2[,"lm.F"] -
                               indexvals["lm.F"])
lRPm.F.se <- sqrt(PotencyFrame2[,"lm.F.se"]^2 +
                  PotencyFrame2[indexchem,"lm.F.se"]^2)
PotencyFrame2[,"RPm.F.lcl"] <- exp(PotencyFrame2[,"lm.F"] -
                               indexvals["lm.F"] -
                                   1.96*lRPm.F.se)
PotencyFrame2[,"RPm.F.ucl"] <- exp(PotencyFrame2[,"lm.F"] -
                               indexvals["lm.F"] +
                                   1.96*lRPm.F.se)
PotencyFrame2[indexchem,c("RPm.F.lcl","RPm.F.ucl")] <- c(NA,NA)
## Males
PotencyFrame2[,"RPm.M"] <- exp(PotencyFrame2[,"lm.M"] -
                               indexvals["lm.M"])
lRPm.M.se <- sqrt(PotencyFrame2[,"lm.M.se"]^2 +
                  indexvals["lm.M.se"]^2)
PotencyFrame2[,"RPm.M.lcl"] <- exp(PotencyFrame2[,"lm.M"] -
                               indexvals["lm.M"] -
                                   1.96*lRPm.M.se)
PotencyFrame2[,"RPm.M.ucl"] <- exp(PotencyFrame2[,"lm.M"] -
                               indexvals["lm.M"] +
                                   1.96*lRPm.M.se)
PotencyFrame2[indexchem,c("RPm.M.lcl","RPm.M.ucl")] <- c(NA,NA)
## Potency based on 'BMD'
## Females
PotencyFrame2[,"RPBMD.F"] <- exp(indexvals["lBMD.F"] -
                               PotencyFrame2[,"lBMD.F"])
lRPBMD.F.se <- sqrt(PotencyFrame2[,"lBMD.F.se"]^2 +
                  indexvals["lBMD.F.se"]^2)
PotencyFrame2[,"RPBMD.F.lcl"] <- exp(indexvals["lBMD.F"] -
                               PotencyFrame2[,"lBMD.F"] -
                                   1.96*lRPBMD.F.se)
PotencyFrame2[,"RPBMD.F.ucl"] <- exp(indexvals["lBMD.F"] -
                               PotencyFrame2[,"lBMD.F"] +
                                   1.96*lRPBMD.F.se)
PotencyFrame2[indexchem,c("RPBMD.F.lcl","RPBMD.F.ucl")] <- c(NA,NA)
## Males
PotencyFrame2[,"RPBMD.M"] <- exp(indexvals["lBMD.M"] -
                               PotencyFrame2[,"lBMD.M"])
lRPBMD.M.se <- sqrt(PotencyFrame2[,"lBMD.M.se"]^2 +
                  indexvals["lBMD.M.se"]^2)
PotencyFrame2[,"RPBMD.M.lcl"] <- exp(indexvals["lBMD.M"] -
                               PotencyFrame2[,"lBMD.M"] -
                                   1.96*lRPBMD.M.se)
PotencyFrame2[,"RPBMD.M.ucl"] <- exp(indexvals["lBMD.M"] -
                               PotencyFrame2[,"lBMD.M"] +
                                   1.96*lRPBMD.M.se)
PotencyFrame2[indexchem,c("RPBMD.M.lcl","RPBMD.M.ucl")] <- c(NA,NA)

save(PotencyFrame2,file="PotencyFrame2.rda")
```

xiii. getBs-etc.R

```
### getBs-etc
###
### Extracts values of B and its se from the model fits in BRAIN-Models-2-save
### as well as estimates of variance components.  Also goes through fits
### in BRAIN-pkfits-2 and does anova on comparable model in BRAIN-Models-2-save

basicmodels <- "BRAIN-Models-2-save"
pkmodels <- "BRAIN-pkfits-2"
Bframe <- data.frame(B.F=numeric(length(Chemicals)),
                     B.F.se=numeric(length(Chemicals)),
                     B.M=numeric(length(Chemicals)),
                     B.M.se=numeric(length(Chemicals)),
                     Respscale=numeric(length(Chemicals)),
                     Dosescale=numeric(length(Chemicals)),
                     mrid.A.sd=numeric(length(Chemicals)),
                     mrid.B.sd=numeric(length(Chemicals)),
                     mrid.m.sd=numeric(length(Chemicals)),
                     set.A.sd=numeric(length(Chemicals)),
                     set.B.sd=numeric(length(Chemicals)),
                     set.m.sd=numeric(length(Chemicals)),
                     row.names=Chemicals)

Bframe[,] <- NA
for (chem in Chemicals) {
  load(file.path(basicmodels,chem))
  tTable <- summary(xx$Fitm)$tTable
  Best <- TRUE
  if ("B.sexF" %in% rownames(tTable)) {
    Bframe[chem,"B.F"] <- tTable["B.sexF",1]
    Bframe[chem,"B.F.se"] <- tTable["B.sexF",2]
    Bframe[chem,"B.M"] <- tTable["B.sexM",1]
    Bframe[chem,"B.M.se"] <- tTable["B.sexM",2]
  } else {
    Bframe[chem,"B.F.se"] <- Bframe[chem,"B.M.se"] <- NA
    Bframe[chem,"B.F"] <- log(Bstart2[chem,"B.F"]/(1 - Bstart2[chem,"B.F"]))
    Bframe[chem,"B.M"] <- log(Bstart2[chem,"B.M"]/(1 - Bstart2[chem,"B.M"]))
    Best <- FALSE
  }
  Bframe[chem,"Dosescale"] <- xx$Dosescale
  Bframe[chem,"Respscale"] <- xx$Respscale

  if (!is.null(xx$Random)) {
    varcomp <- lapply(lapply(xx$Fitm$modelStruct$reStruct,as.matrix),
                      function(x) x*xx$Fitm$sigma^2)
    if ("mrid" %in% names(varcomp)) {
      if ("A.(Intercept)" %in% rownames(varcomp$mrid))
        Bframe[chem,"mrid.A.sd"] <- sqrt(varcomp$mrid["A.(Intercept)",
                                                      "A.(Intercept)"])
      Bframe[chem,"mrid.m.sd"] <- sqrt(varcomp$mrid["m.(Intercept)",
                                                    "m.(Intercept)"])
      if (Best) {
        Bframe[chem,"mrid.B.sd"] <- sqrt(varcomp$mrid["B.(Intercept)",
                                                      "B.(Intercept)"])
      }
    }
    if ("set" %in% names(varcomp)) {
      if ("A.(Intercept)" %in% rownames(varcomp$mrid))
        Bframe[chem,"set.A.sd"] <- sqrt(varcomp$set["A.(Intercept)",
                                                    "A.(Intercept)"])
      Bframe[chem,"set.m.sd"] <- sqrt(varcomp$set["m.(Intercept)",
                                                  "m.(Intercept)"])
      if (Best) {
        Bframe[chem,"set.B.sd"] <- sqrt(varcomp$set["B.(Intercept)",
                                                    "B.(Intercept)"])
      }
    }
  } else {
    Bframe[chem,c("mrid.A.sd","mrid.B.sd","mrid.m.sd",
                  "set.A.sd","set.B.sd","set.m.sd")] <- NA
  }
}

### Make a .csv file of Pb estimates and confidence limits, males and
### females:

Btable <- as.data.frame(matrix(as.numeric(NA),nrow=nrow(Bframe),ncol=6,
                 dimnames=list(rownames(Bframe),
                   c("PB.F","PB.F.lcl","PB.F.ucl",
                     "PB.M","PB.M.lcl","PB.M.ucl"))))
Btable[,"PB.F"] <- 1/(1 + exp(-Bframe[,"B.F"]))
Btable[,"PB.M"] <- 1/(1 + exp(-Bframe[,"B.M"]))
Btable[,"PB.F.lcl"] <- 1/(1 + exp(-(Bframe[,"B.F"]+qnorm(0.025)*Bframe[,"B.F.se"])))
Btable[,"PB.F.ucl"] <- 1/(1 + exp(-(Bframe[,"B.F"]+qnorm(0.975)*Bframe[,"B.F.se"])))
Btable[,"PB.M.lcl"] <- 1/(1 + exp(-(Bframe[,"B.M"]+qnorm(0.025)*Bframe[,"B.M.se"])))
Btable[,"PB.M.ucl"] <- 1/(1 + exp(-(Bframe[,"B.M"]+qnorm(0.975)*Bframe[,"B.M.se"])))

### Write them out
write.table(Btable,file="Btable.cvs",sep=",",col.names=NA)

### Plot 'em by activity
Action <- read.csv("active.txt",row.names=1)
Btable$Action <- Action[rownames(Btable),"Oxon"]
sindx <- order(Btable$PB.F)
Btable <- Btable[sindx,]
indx1 <- which(Btable$Action=="Direct")
indx2 <- which(Btable$Action=="Indirect")
Btable$Index <- c(indx1,indx2)
postscript(file="Bplot.eps",onefile=FALSE,horizontal=FALSE,pointsize=12)
omar <- par("mar")
par(mar=omar + c(7,0,0,0),pty="s")
indx <- Btable$Index
x.F <- (1:29) - 0.15
x.M <- (1:29) + 0.15
plotCI(c(x.F,x.M),c(Btable[indx,"PB.F"],Btable[indx,"PB.M"]),
       aui=c(Btable[indx,"PB.F.ucl"],Btable[indx,"PB.M.ucl"]),
       ali=c(Btable[indx,"PB.F.lcl"],Btable[indx,"PB.M.lcl"]),
       lwd=2,xaxt="n",xlab="",ylab=expression(P[B]),col=rep(c("red","blue"),
                                 c(length(x.F),length(x.M))),
       scol=rep(c("red","blue"),c(length(x.F),length(x.M))),
       gap=0.005,
       ylim=c(0,0.8),las=1,cex=0.75,sfrac=0.005)
axis(side=1,at=1:29,labels=row.names(Btable)[indx],las=3)
abline(v=length(indx1)+0.5,lty=3)
text(c((1+length(indx1))/2,length(indx1) + (1+length(indx2))/2),
     c(0.7,0.7),
     labels=c("Direct Acting","Require Activation"),adj=c(0.5,0.5),
     cex=1.2)

dev.off()


### Test: are the B's homogeneous across chemicals (within sex)?

### The basis for this test is the idea that the estimates of B should
### be approximately normal, but with different standard errors.  Only
### B's estimated within the model are included.

### Females First:

se.F <- Bframe[,"B.F.se"]
B.F <- Bframe[!is.na(se.F),"B.F"]
se.F <- se.F[!is.na(se.F)]

## estimate the mean: use the weighted average of B.F
## wts = 1/se.F^2

B.F.mn <- weighted.mean(B.F,1/se.F^2)

Q <- sum(((B.F - B.F.mn)/se.F)^2)

cat("Females:\n")
cat(paste("Significance of heterogeneity among B estimates:\n X^2 = ",
          signif(Q,4),
          "with",length(B.F)-1,"df.","P =",
          signif(pchisq(Q,length(B.F)-1,lower.tail=FALSE),2),"\n"))

### Now Males:
se.M <- Bframe[,"B.M.se"]
B.M <- Bframe[!is.na(se.M),"B.M"]
se.M <- se.M[!is.na(se.M)]

## estimate the mean: use the weighted average of B.M
## wts = 1/se.M^2

B.M.mn <- weighted.mean(B.M,1/se.M^2)

Q <- sum(((B.M - B.M.mn)/se.M)^2)

cat("\nMales:\n")
cat(paste("Significance of heterogeneity among B estimates:\n X^2 = ",
          signif(Q,4),
          "with",length(B.M)-1,"df.","P =",
          signif(pchisq(Q,length(B.M)-1,lower.tail=FALSE),2),"\n"))
```

xiv. plotAppendixFigs.R

```
### plotAppendixFigs.R
###
### plots all the appendix figures: for each chemical, plot on a 3 X 2
### grid.  Each page is in the file 'Chemical'-pg-xx.eps.
require(akima)

## Some directories:
Modelsdir <- "BRAIN-Models-2-save"
pkModelsdir <- "BRAIN-pkfits-2"
Figuresdir <- "AppendixFigs"
Bproflikdir <- "BRAIN-Models-BPL"
pkproflikdir <- "BRAIN-pkgrids"


for (chem in Chemicals) {
  postscript(file=file.path(Figuresdir,paste(chem,"-pg-%03d.eps",sep="")),
             onefile=FALSE,horizontal=FALSE,pointsize=12)
  par(mfrow=c(3,2))
  cat(paste("-------\n",chem,"\n\n"))
### load the model fit
  load(file.path(Modelsdir,chem))

  seldata <- xx$Data
  seldata$block <- factor(seldata$block)
  seldata$mrid <- factor(seldata$mrid)
  seldata$Unit <- factor(seldata$chunit)
  seldata$s.U <- factor(interaction(seldata$sex, seldata$Unit, drop=TRUE))
  B.est <- "B.sexF" %in% if (inherits(xx$Fitm,"nlme"))
    names(xx$Fitm$coefficients$fixed) else names(xx$Fitm$coefficients)
### Set up the dataframe to compute residuals, etc.

  seldata$Dose.scaled <- seldata$dose/xx$Dosescale
  seldata$set <- factor(paste(seldata$mrid,seldata$time,seldata$duplicate,sep=":"))
  seldata2  <- seldata
  seldata2$Dose.scaled[] <- 0

### for each unique level in s.U, plot response versus dose, and predicted
### response versus dose.  Plot both sexes on the same plot (red for females,
### blue for males).  Each unit gets its own plot.
  for (Un in levels(seldata$Unit)) {
    yy <- seldata$chei[seldata$Unit == Un]
    x <- seldata$dose[seldata$Unit == Un]
    xpred <- seq(0,max(x),length=25)
    plot(x,yy,xlab="Dose (mg/kg/day)",ylab=paste("AChE (",Un,")",sep=""),
         type="n",main=Un,ylim=range(0,yy))
    for (sex in c("F","M")) {
      yy <- seldata$chei[seldata$Unit == Un & seldata$sex == sex]
      if (length(yy) == 0) next
      x <- seldata$dose[seldata$Unit == Un & seldata$sex == sex]
      points(x,yy,col=c(F="red",M="blue")[sex])
      newdata <- data.frame(s.U = factor(rep(paste(sex,Un,sep="."),
                              length(xpred)),levels=levels(seldata$s.U)),
                            Dose.scaled = xpred/xx$Dosescale,
                            sex = factor(rep(sex,length(xpred)),
                              levels=c("F","M")))
      if (inherits(xx$Fitm,"nlme"))
        yypred <- myPredict(xx$Fitm,newdata=newdata,level=0)*xx$Respscale
      else
        yypred <- predict(xx$Fitm,newdata=newdata)*xx$Respscale
      lines(spline(xpred,yypred),col=c(F="red",M="blue")[sex])
      ## If there are multiple mrids, plot their curves as dotted lines
      if (inherits(xx$Fitm, "nlme")) {
        tmp <- seldata[seldata$Unit == Un & seldata$sex == sex,]
        if (length(unique(tmp$mrid)) > 1) {
          for (mrid in sort(unique(as.character(tmp$mrid)))) {
            newdata <- data.frame(s.U = factor(rep(paste(sex,Un,sep="."),
                                    length(xpred)),levels=levels(seldata$s.U)),
                                  Dose.scaled = xpred/xx$Dosescale,
                                  sex = factor(rep(sex,length(xpred)),
                                    levels=c("F","M")),
                                  mrid = factor(rep(mrid, length(xpred))))
            yypred <- myPredict(xx$Fitm,newdata=newdata,level=1)*xx$Respscale
            lines(spline(xpred,yypred),col=c(F="red",M="blue")[sex],lty=2)
          }
        }
      }
    }
  }
### Plots, etc. for Fitm
  if (inherits(xx$Fitm,"nlme")) {
    lvl <- switch(length(xx$Random[[3]][[3]]),1,2,2)
    preddata <- myPredict(xx$Fitm,newdata=seldata,
                          level=lvl)
    contdata <- myPredict(xx$Fitm,newdata=seldata2,
                          level=lvl)
  } else {
    preddata <- predict(xx$Fitm,newdata=seldata)
    contdata <- predict(xx$Fitm,newdata=seldata2)
  }
  Inh <- 1 - preddata/contdata
  resids <- sqrt(seldata$n)*(seldata$chei/xx$Respscale - preddata)/
              (xx$Fitm$sigma*preddata^0.5)
  plot(resids ~ Inh,ylab="Scaled Residuals",
       xlab="Predicted Fraction Inhibition",
       main="Scaled Residuals versus Predicted Inhibition")
  abline(h=0,lty=2)
### Profile likelihood for Bs:
  ## save the model stuff:
  xxsave <- xx
  load(file.path(Bproflikdir,chem))
  Bgrid <- xx$Bgrid
  Bgrid$LL <- xx$LL
### Ethoprop inexplicably dies with an out-of-memory error before it is done
  if (chem == "ETHOPROP")Bgrid$LL[Bgrid$LL == 0] <- NA
  Bgrid <- na.omit(Bgrid)
  out <- interp(Bgrid$B.F,Bgrid$B.M,Bgrid$LL,seq(0,1,length=40),
              seq(0,1,length=40))
  image(out,xlab=expression(paste("Female ", P[B])),
        ylab=expression(paste("Male ", P[B])),
        main=expression(paste("Profile Likelihood for ", P[B])))
  points(Bgrid$B.F,Bgrid$B.M,pch=3,cex=0.5)

  if (B.est) {
    if (inherits(xxsave$Fitm,"nlme")) {
      B.F <- xxsave$Fitm$coefficients$fixed["B.sexF"]
      B.M <- xxsave$Fitm$coefficients$fixed["B.sexM"]
    } else {
      B.F <- xxsave$Fitm$coefficients["B.sexF"]
      B.M <- xxsave$Fitm$coefficients["B.sexM"]
    }
    B.F <- 1/(1 + exp(-B.F))
    B.M <- 1/(1 + exp(-B.M))
    points(B.F,B.M)
  }
### Profile likelihood for S and D for pkmodel
  load(file.path(pkproflikdir,chem))
  LLgrid <- xx$LLgrid
  LLgrid <- na.omit(LLgrid)
  if (length(LLgrid$LL) == 0) {
    plot(c(0,1),c(0,1),type="n",axes=FALSE, xlab="", ylab="")
    text(0.5,0.5,paste(chem,"no fits"),adj=0.5)
  } else {
    out <- try(interp(LLgrid$S,LLgrid$D,LLgrid$LL,
                      seq(min(LLgrid$S), max(LLgrid$S),length=40),
                      seq(min(LLgrid$D), max(LLgrid$D),length=40)))
    if (!inherits(out, "try-error")) {
      res <- try(image(out,xlab="S",ylab="D",
                       main="Profile Likelihood for D and S"))
      if (inherits(res, "try-error")) {
        plot(c(0,1),c(0,1),type="n",axes=FALSE, xlab="", ylab="")
        text(0.5,0.5,paste(chem,"not enough fits"),adj=0.5)
      } else {
        points(LLgrid$S,LLgrid$D,pch=3,cex=0.5)
      }
    } else {
      plot(c(0,1),c(0,1),type="n",axes=FALSE, xlab="", ylab="")
      text(0.5,0.5,paste(chem,"not enough fits"),adj=0.5)
    }
  }
### If we have a fit, load the file, plot S and D on the above figure,
### and then plot dose-responses and residuals versus predicted inhibition
  if (file.exists(file.path(pkModelsdir,chem))) {
    load(file.path(pkModelsdir,chem))
    if (!inherits(xx$Fitpk,"try-error")) {
      lD <- if (inherits(xx$Fitpk,"nlme")) xx$Fitpk$coefficients$fixed["D"] else
              xx$Fitpk$coefficients["D"]
      lS <- if (SDstart[chem,"S"] < 0.0001) -15 else
      log(SDstart[chem,"S"])
      points(exp(lS),exp(lD))


      seldata <- xx$Data
      seldata$block <- factor(seldata$block)
      seldata$mrid <- factor(seldata$mrid)
      seldata$Unit <- factor(seldata$chunit)
      seldata$s.U <- factor(interaction(seldata$sex, seldata$Unit, drop=TRUE))

### Set up the dataframe to compute residuals, etc.

      seldata$Dose.scaled <- seldata$dose/xx$Dosescale
      seldata$set <-
factor(paste(seldata$mrid,seldata$time,seldata$duplicate,sep=":"))
      seldata2  <- seldata
      seldata2$Dose.scaled[] <- 0

### for each unique level in s.U, plot response versus dose, and predicted
### response versus dose.  Plot both sexes on the same plot (red for females,
### blue for males).  Each unit gets its own plot.
      for (Un in levels(seldata$Unit)) {
        yy <- seldata$chei[seldata$Unit == Un]
        x <- seldata$dose[seldata$Unit == Un]
        xpred <- seq(0,max(x),length=101)
        plot(x,yy,xlab="Dose (mg/kg/day)",ylab=paste("AChE (",Un,")",sep=""),
             type="n",main=Un,ylim=range(0,yy))
        for (sex in c("F","M")) {
          yy <- seldata$chei[seldata$Unit == Un & seldata$sex == sex]
          if (length(yy) == 0) next
          x <- seldata$dose[seldata$Unit == Un & seldata$sex == sex]
          points(x,yy,col=c(F="red",M="blue")[sex])
          newdata <- data.frame(s.U = factor(rep(paste(sex,Un,sep="."),
                                  length(xpred)),levels=levels(seldata$s.U)),
                                Dose.scaled = xpred/xx$Dosescale,
                                sex = factor(rep(sex,length(xpred)),
                                  levels=c("F","M")))
          if (inherits(xx$Fitpk,"nlme"))
            yypred <- mypkPredict(xx$Fitpk,newdata=newdata,
                                  level=0)*xx$Respscale
          else
            yypred <- predict(xx$Fitpk,newdata=newdata)*xx$Respscale
          lines(xpred,yypred,col=c(F="red",M="blue")[sex])
          ## If there are multiple mrids, plot their curves as dotted lines
          if (inherits(xx$Fitpk, "nlme")) {
            tmp <- seldata[seldata$Unit == Un & seldata$sex == sex,]
            if (length(unique(tmp$mrid)) > 1) {
              for (mrid in sort(unique(as.character(tmp$mrid)))) {
                newdata <- data.frame(s.U = factor(rep(paste(sex,Un,sep="."),
                                          length(xpred)),
                                          levels=levels(seldata$s.U)),
                                        Dose.scaled = xpred/xx$Dosescale,
                                        sex = factor(rep(sex,length(xpred)),
                                          levels=c("F","M")),
                                        mrid = factor(rep(mrid, length(xpred))))
                yypred <- mypkPredict(xx$Fitpk,newdata=newdata,
                                        level=1)*xx$Respscale
                lines(xpred,yypred,col=c(F="red",M="blue")[sex],lty=2)
              }
            }
          }
        }
      } ## matches for (Un in levels(seldata$Unit))
### Plots, etc. for Fitpk
      if (inherits(xx$Fitpk,"nlme")) {
        lvl <- switch(length(xx$Random[[3]][[3]]),1,2,2)
        preddata <- mypkPredict(xx$Fitpk,newdata=seldata,
                                level=lvl)
        contdata <- mypkPredict(xx$Fitpk,newdata=seldata2,
                                level=lvl)

      } else {
        preddata <- predict(xx$Fitpk,newdata=seldata)
        contdata <- predict(xx$Fitpk,newdata=seldata2)
      }
      Inh <- 1 - preddata/contdata
      resids <- sqrt(seldata$n)*(seldata$chei/xx$Respscale - preddata)/
                  (xx$Fitpk$sigma*preddata^0.5)
      plot(resids ~ Inh,ylab="Scaled Residuals",
           xlab="Predicted Fraction Inhibition",
           main="Residuals from Model w/Low Dose Curvature")
      abline(h=0,lty=2)
    }
  }
### Close the current chemical
  graphics.off()
}
```

xv. printParameters.R

```
### printParameters.R
### create data frames to contain the model parameters, and then dump them
### out as .csv files.
### Use the fits for the basic model in

basicdir <- "BRAIN-Models-2-save"

### When they exist, replace these with parameters from fits in

expanddir <- "BRAIN-pkfits-2"

### Create Table 1: Log Background Levels.  Has one row per chemical,
### two columns for each unit X sex combination (one for value, one for
### se).

Table1 <- matrix(NA,nrow=29,ncol=28,
                 dimnames=list(Chemicals,
                   c("U/G.F.val",
                     "U/G.F.se",
                     "U/G.M.val",
                     "U/G.M.se",
                     "U/G1.F.val",
                     "U/G1.F.se",
                     "U/G1.M.val",
                     "U/G1.M.se",
                     "U/G2.F.val",
                     "U/G2.F.se",
                     "U/G2.M.val",
                     "U/G2.M.se",
                     "U/G3.F.val",
                     "U/G3.F.se",
                     "U/G3.M.val",
                     "U/G3.M.se",
                     "PHCHANGE.F.val",
                     "PHCHANGE.F.se",
                     "PHCHANGE.M.val",
                     "PHCHANGE.M.se",
                     "U/L.F.val",
                     "U/L.F.se",
                     "U/L.M.val",
                     "U/L.M.se",
                     "U/mL.F.val",
                     "U/mL.F.se",
                     "U/mL.M.val",
                     "U/mL.M.se")))

### Table 2: all the other fixed parameters
Table2 <- matrix(NA,nrow=29,ncol=11,
                 dimnames=list(Chemicals,
                   c("lm.F.val",
                     "lm.F.se",
                     "lm.M.val",
                     "lm.M.se",
                     "tB.F.val",
                     "tB.F.se",
                     "tB.M.val",
                     "tB.M.se",
                     "lS",
                     "lD.val",
                     "lD.se")))

### Table 3: log BMD, w/standard errors, BMD w/95% CI, rel potency w/95% CI.
Table3 <- matrix(NA, nrow=29, ncol=16,
                 dimnames=list(Chemicals,
                   c("lBMD.F.val",
                     "lBMD.F.se",
                     "lBMD.M.val",
                     "lBMD.M.se",
                     "BMD.F",
                     "BMD.F.lcl",
                     "BMD.F.ucl",
                     "BMD.M",
                     "BMD.M.lcl",
                     "BMD.M.ucl",
                     "relpot.F",
                     "relpot.F.lcl",
                     "relpot.F.ucl",
                     "relpot.M",
                     "relpot.M.lcl",
                     "relpot.M.ucl")))

### We're going to duplicate some effort here, since some of these values
### have already been tabulated.

for (chem in Chemicals) {
  ##  load the model fits and determine whether there is an expanded model
  ##  fit as well (value of doexpand)
  load(file.path(basicdir,chem))
  xsave <- xx
  if (file.exists(file.path(expanddir,chem))) {
    load(file.path(expanddir,chem))
    if (!inherits(xx$Fitpk,"try-error")) {
      doexpand <- TRUE
      Model <- xx$Fitpk
    } else {
      xx <- xsave
      rm(xsave)
      Model <- xx$Fitm
      doexpand <- FALSE
    }
  } else {
    xx <- xsave
    rm(xsave)
    doexpand <- FALSE
    Model <- xx$Fitm
  }
  Dosescale <- xx$Dosescale
  Respscale <- xx$Respscale
  ## Fill in Table 1. 'Model' has the model I am going to use; Dosescale and
  ## Respscale the dose and response scales, respectively.  Need to add back
  ## log(Respscale) to the A parameter estimates (se's do not change)

  tTable <- summary(Model)$tTable

  ## Which rows are for the A.S.U term?
  indx <- grep("A\\.s\\.U",rownames(tTable))
  ## Get the base for the column names in Table 1
  basenames <- sapply(strsplit(gsub("^A\\.s\\.U","",rownames(tTable)[indx]),
                               "\\."),
                      function(x) paste(x[2],x[1],sep="."))
  Table1[chem,paste(basenames,"val",sep=".")] <-
    tTable[indx,1] + log(Respscale)
  Table1[chem,paste(basenames,"se",sep=".")] <- tTable[indx,2]

  ## Fill in Table 2
  ## First lm; this is always estimated

  indx <- grep("^m\\.",rownames(tTable))
  basenames <- paste("l",sub("sex","",rownames(tTable)[indx]),sep="")
  Table2[chem,paste(basenames,"val",sep=".")] <-
    tTable[indx,1] - log(Dosescale)
  Table2[chem,paste(basenames,"se",sep=".")] <- tTable[indx,2]
  ## Next tB; this may be 'fixed'.  If so, it does not have a standard
  ## error.

  indx <- grep("^B\\.",rownames(tTable))
  if (length(indx) > 0) {
    basenames <- paste("t",sub("sex","",rownames(tTable)[indx]),sep="")
    Table2[chem,paste(basenames,"val",sep=".")] <- tTable[indx,1]
    Table2[chem,paste(basenames,"se",sep=".")] <- tTable[indx,2]
  } else {
    vals <- eval(Model$call$model[[3]][["fixed"]][["B"]])
    Table2[chem,c("tB.F.val","tB.M.val")] <- vals[c("F","M")]
  }
  if (doexpand) {
    ## Get lS: always fixed if we fit this model
    vals <- eval(Model$call$model[[3]][["fixed"]][["S"]])
    Table2[chem,"lS"] <- c(vals["F"])

    ## Get lD: always estimated if we fit this model
    Table2[chem,c("lD.val","lD.se")] <- tTable["D",1:2]
  }
  ## Now fill in all the stuff about BMD and lBMD; potencies we handle
  ## after all the chemicals have been done.
  if (doexpand) {
    tmp <- pkexpSlBMD.se(xx,BMR=0.10)
    Table3[chem,c("lBMD.F.val","lBMD.M.val")] <- c(tmp$lBMD[c("F","M")])
    Table3[chem,c("lBMD.F.se","lBMD.M.se")] <- c(tmp$lBMD.se[c("F","M")])
  } else {
    ## Use xx$FitBMD
    tTable <- summary(xx$FitBMD)$tTable
    Table3[chem,c("lBMD.F.val","lBMD.M.val")] <-
      tTable[c("BMD.sexF","BMD.sexM"),1] + log(xx$Dosescale)
    Table3[chem,c("lBMD.F.se","lBMD.M.se")] <-
      tTable[c("BMD.sexF","BMD.sexM"),2]
  }
}

## Finish off Table3

## BMD and CIs

Table3[,c("BMD.F","BMD.M")] <- exp(Table3[,c("lBMD.F.val","lBMD.M.val")])
Table3[,c("BMD.F.lcl","BMD.F.ucl")] <-
  exp(Table3[,"lBMD.F.val"] + cbind(Table3[,"lBMD.F.se"]*qnorm(0.025),
                                    Table3[,"lBMD.F.se"]*qnorm(0.975)))
Table3[,c("BMD.M.lcl","BMD.M.ucl")] <-
  exp(Table3[,"lBMD.M.val"] + cbind(Table3[,"lBMD.M.se"]*qnorm(0.025),
                                    Table3[,"lBMD.M.se"]*qnorm(0.975)))

## Relative Potency
ic <- "METHAMIDOPHOS"

Table3[,"relpot.F"] <- exp(Table3[ic,"lBMD.F.val"] - Table3[,"lBMD.F.val"])
se <- sqrt(Table3[,"lBMD.F.se"]^2 + Table3[ic,"lBMD.F.se"]^2)
Table3[,"relpot.F.lcl"] <- exp(Table3[ic,"lBMD.F.val"] -
                               Table3[,"lBMD.F.val"] +
                               qnorm(0.025)*se)
Table3[,"relpot.F.ucl"] <- exp(Table3[ic,"lBMD.F.val"] -
                               Table3[,"lBMD.F.val"] +
                               qnorm(0.975)*se)

Table3[,"relpot.M"] <- exp(Table3[ic,"lBMD.M.val"] - Table3[,"lBMD.M.val"])
se <- sqrt(Table3[,"lBMD.M.se"]^2 + Table3[ic,"lBMD.M.se"]^2)
Table3[,"relpot.M.lcl"] <- exp(Table3[ic,"lBMD.M.val"] -
                               Table3[,"lBMD.M.val"] +
                               qnorm(0.025)*se)
Table3[,"relpot.M.ucl"] <- exp(Table3[ic,"lBMD.M.val"] -
                               Table3[,"lBMD.M.val"] +
                               qnorm(0.975)*se)
```

```
Table3[ic,c("relpot.F.lcl","relpot.F.ucl","relpot.M.lcl","relpot.M.ucl")] <- NA

### Now dump them.  First, dump all three tables as .csv files

write.table(Table1,file="Table1.csv",sep=",",col.names=NA)
write.table(Table2,file="Table2.csv",sep=",",col.names=NA)
write.table(Table3,file="Table3.csv",sep=",",col.names=NA)
```