Individual Bayesian Information matrix for predicting estimation error and shrinkage of individual parameters accounting for data below the limit of quantification – Supplementary material

Thi Huyen Tram NGUYEN[*], Thu Thuy NGUYEN[*], France MENTRE

INSERM, IAME, UMR 1137, F-75018 Paris, France; University Paris Diderot, Sorbonne Paris Cité, F-75018 Paris, France

([*]) THT Nguyen and TT Nguyen contributed equally to the manuscript

Corresponding author:
Thu Thuy Nguyen
INSERM, IAME, UMR 1137 - University Paris Diderot
16 rue Henri Huchard
75018 Paris, France
Tel: 00 33 1 57 27 75 35
Email: thu-thuy.nguyen@inserm.fr

R script: The present R code implements the likelihood for the studied pharmacokinetic/viral kinetic model written using differential equations, accounting for the contributions of observed and censored data (data below the limit of quantification) respectively. The package *marqLevAlg* was used for likelihood maximization.

```
library(deSolve)
library(reshape2)

#ODE model
formED <- function(t,y,p) {
  ka <- p[1]
  ke <- p[2]
  V <- p[3]
  EC50 <- p[4]
  VL0 <- 10^p[5]
  delta <- p[6]
  c <- p[7]
  tau <- 7
  beta <- 5.6e-7
  pp <- 10
  nhill <- 1
  dose <- 180
  y0.func<-function(ka,dose,n,tau,t){
    if(n==0){return(dose*exp(-ka*t))}
    else{return(dose*exp(-ka*(t-n*tau))+y0.func(ka,dose,n-1,tau,t))}}
  n<-t%/%tau
  y0<-y0.func(ka,dose,n,tau,t)
  Tc <- delta*c/beta/pp
  dy1 <- ka*y0-ke*y[1]
  Cc <- y[1]/V
  epsilon <- Cc^nhill/(EC50^nhill+Cc^nhill)
  dy2 <- beta*y[3]*Tc-delta*y[2]
  dy3 <- (1-epsilon)*pp*y[2]-c*y[3]
  return(list(c(dy1,dy2,dy3),c(Cc,log10(y[3]))))}


RtolEQ<-1e-08
AtolEQ<-1e-16
#Hmax<-Inf

# Data simulation

beta<-c(0.9,0.15,12,0.15,6,0.2,6)
sigmainterA <- 0.5
sigmainterB <- 0.15

parameters <- c("ka","ke","V","EC50","VL0","delta","c")
t <- c(0, 1, 3, 5, 7, 14, 28)
condinit <- expression(c(0,c*10^VL0/10,10^VL0))
p <- length(beta)
for (i in 1:p) {assign(parameters[i],beta[i]) }
cond<-eval(condinit)

set.seed(688690)
sigmaPK <- rnorm(length(t),0,sigmainterA)
sigmaPD <- rnorm(length(t),0,sigmainterB)
obs <- as.data.frame(lsoda(cond,t,formED,beta,rtol=RtolEQ,atol=AtolEQ))
obs <- obs[,-c(2:4)]
obs[,2] <- obs[,2]+sigmaPK
obs[,3] <- obs[,3]+sigmaPD
obs <- as.data.frame(obs)

colnames(obs) <- c("time","1","2")
obs <- melt(obs,id.var=c("time"),variable.name="Type",value.name="Y")
```

```
obs_type1 <- obs[which(obs$Type==1 & obs$time %in% c(1, 3, 5, 7, 14)),]
obs_type2 <- obs[which(obs$Type==2 & obs$time %in% c(0, 1, 3, 5, 7, 14,
28)),]
obs <- as.data.frame(rbind(obs_type1,obs_type2))

obs$cens <- 0
obs$cens[obs$Type==2 & obs$Y <= log10(100)] <- 1


# Loglikelihood function

lnl2 <- function(parms){
  #print(parms)
  t <- obs$time
  t <- sort(unique(t))
  parms <- abs(parms)
  ka <- parms[1]
  ke <- parms[2]
  V <- parms[3]
  EC50 <- parms[4]
  VL0 <- parms[5]
  delta <- parms[6]
  c <- parms[7]
  sigmainter <- parms[8]
  sigmainter2 <- parms[9]
  cinit <- eval(condinit)
  out <-
lsoda(y=cinit,times=t,func=formED,parms=parms,rtol=RtolEQ,atol=AtolEQ)[,-
c(2:4)]
  pred <- as.data.frame(out)
  colnames(pred) <- c("time","Y1","Y2")

  L<-0
  for (j in obs$Type) {
    tps <- obs$time[obs$Type==j]
    Y <- obs$Y[obs$Type==j]
    cens <- obs$cens[obs$Type==j]
    if (j==1) {
      sigmaexp <- expression(sigmainter)
      predj <- pred$Y1[pred$time %in% tps]
    } else {
      sigmaexp <- expression(sigmainter2)
      predj <- pred$Y2[pred$time %in% tps]
    }
    for (i in 1:length(tps)){
      sigma <- eval(sigmaexp)
      if (cens[i]==0) {
        L <- L+log(((1/(sqrt(2*pi*sigma^2)))*exp(-(Y[i]-
predj[i])^2/(2*sigma^2))))}
      if (cens[i] == 1) {
        L <- L+log((pnorm((Y[i]-predj[i])/sigma)))}
    }}
  return(-2*L)}

start<-c(0.9,0.15,12,0.15,6,0.2,6,0.5,0.15)

library(marqLevAlg)
marqLevAlg(b=start,fn=lnl2)
```