

Package ‘nlmeODE’

July 2, 2014

Version 1.1

Date 2012/10/27

Title Non-linear mixed-effects modelling in nlme using differential equations

Author Christoffer W. Tornoe <ctornoe@gmail.com>

Maintainer Christoffer W. Tornoe <ctornoe@gmail.com>

Depends R (>= 2.6.0), deSolve, nlme, lattice

Description This package combines the odesolve and nlme packages for mixed-effects modelling using differential equations.

License LGPL (>= 2.1)

URL <http://www.cran.r-project.org>

Repository CRAN

Date/Publication 2012-10-27 21:08:52

NeedsCompilation no

R topics documented:

nlmeODE	2
PKPDmodels	3

Index	16
--------------	-----------

nlmeODE	<i>Non-linear mixed-effects modelling in nlme using differential equations</i>
---------	--

Description

This package combines the deSolve and nlme packages for mixed-effects modelling.

Usage

```
nlmeODE(model, data, LogParms, JAC, SEQ, rtol, atol, tcrit, hmin, hmax)
```

Arguments

model	A list including the following elements: DiffEq (A list of formulas containing the ODE's for the system in the same order as the compartment numbers, i.e. formula <i>i</i> is the ODE for compartment <i>i</i>), ObsEq (A formula specifying which state is observed along with possible scaling parameters), Parm (A vector with the names of the parameters used in DiffEq followed by the parameters in ObsEq and the initial state parameters), States (A vector with the names of the states in DiffEq), Init (A logical vector with the same length as States specifying whether initial state estimates should be obtained for the particular state when TRUE)
data	groupedData object with a formula specifying which columns are the dependent and independent variables, and grouping factor. Optional columns in the groupedData object are the dose Dose, dosing compartment Cmt, rate of infusion Rate, covariate measurements, etc.
LogParms	If TRUE, the parameters are reparameterized in terms of the logarithm of the parameters. Default is TRUE.
JAC	If TRUE, the Jacobian of the system of ODE's is computed and passed to the ODE solver. In some circumstances, supplying the Jacobian can speed up the computations if the system is stiff. Default is FALSE.
SEQ	A logical value, that when TRUE, adds a gradient attribute to the returned value calculated by simultaneous solution of the sensitivity equations associated with the system of ODE's. Default is FALSE.
rtol	Relative error tolerance for lsoda. Default is 1E-4.
atol	Absolute error tolerance for lsoda. Default is 1E-4.
tcrit	Time beyond which the integration should not proceed. Default is NULL.
hmin	Minimum value of the integration stepsize. Default is 0.
hmax	Maximum value of the integration stepsize. Default is Inf.

Value

A function compatible with nlme estimation for systems of ordinary differential equations (ODE's).

Note

For examples of commonly used PK/PD models see [PKPDmodels](#).

Author(s)

Christoffer W. Tornoe <<ctornoe@gmail.com>>

References

Tornoe, C. W. et al. (2004a) "Non-linear mixed-effects pharmacokinetic/pharmacodynamic modelling in NLME using differential equations", *Computer Methods and Programs in Biomedicine*, 76(1), 31-40 and Tornoe, C. W. et al. (2004b) "Pharmacokinetic/Pharmacodynamic Modelling of GnRH Antagonist Degarelix: A Comparison of the Non-linear Mixed-Effects Programs NONMEM and NLME", *Journal of Pharmacokinetics and Pharmacodynamics*, 31(6), 441-461.

See Also

[nlme](#), [lsoda](#)

PKPDmodels

Library of PK/PD models

Description

Library of common PK/PD models.

Examples

```
#####
### Pharmacokinetics of Theophylline ###
#####

data(Theoph)

TheophODE <- Theoph
TheophODE$Dose[TheophODE$Time!=0] <- 0
TheophODE$Cmt <- rep(1,dim(TheophODE)[1])

OneComp <- list(DiffEq=list(
  dy1dt = ~ -ka*y1 ,
  dy2dt = ~ ka*y1-ke*y2),
  ObsEq=list(
    c1 = ~ 0,
    c2 = ~ y2/CL*ke),
  Params=c("ka","ke","CL"),
  States=c("y1","y2"),
  Init=list(0,0))
```

```

TheophModel <- nlmeODE(OneComp,TheophODE)

## Not run:
Theoph.nlme <- nlme(conc ~ TheophModel(ka,ke,CL,Time,Subject),
  data = TheophODE, fixed=ka+ke+CL~1, random = pdDiag(ka+CL~1),
  start=c(ka=0.5,ke=-2.5,CL=-3.2),
  control=list(returnObject=TRUE,msVerbose=TRUE),
  verbose=TRUE)

plot(augPred(Theoph.nlme,level=0:1))

## End(Not run)

#####
### Pharmacokinetics of Indomethacine ###
#####

data(Indometh)

TwoComp <- list(DiffEq=list(
  dy1dt = ~ -(k12+k10)*y1+k21*y2 ,
  dy2dt = ~ -k21*y2 + k12*y1),
  ObsEq=list(
    c1 = ~ y1,
    c2 = ~ 0),
  States=c("y1","y2"),
  Params=c("k12","k21","k10","start"),
  Init=list("start",0))

IndomethModel <- nlmeODE(TwoComp,Indometh)

## Not run:
Indometh.nlme <- nlme(conc ~ IndomethModel(k12,k21,k10,start,time,Subject),
  data = Indometh, fixed=k12+k21+k10+start~1, random = pdDiag(start+k12+k10~1),
  start=c(k12=-0.05,k21=-0.15,k10=-0.10,start=0.70),
  control=list(msVerbose=TRUE),
  verbose=TRUE)

plot(augPred(Indometh.nlme,level=0:1))

## End(Not run)

#####
### Absorption model with estimation of time/rate of infusion ###
#####

OneCompAbs <- list(DiffEq=list(
  dA1dt = ~ -ka*A1,
  dA2dt = ~ ka*A1 - CL/V1*A2),
  ObsEq=list(
    SC= ~0,
    C = ~ A2/V1),
  States=c("A1","A2"),

```

```

      Parms=c("ka", "CL", "V1", "F1"),
      Init=list(0,0))

ID   <- rep(seq(1:18),each=11)
Time <- rep(seq(0,100,by=10),18)
Dose <- c(rep(c(100,0,0,100,0,0,0,0,0,0),6),rep(c(100,0,0,0,0,0,100,0,0,0),6),
rep(c(100,0,0,0,0,0,0,0,0,0),6))
Rate <- c(rep(rep(0,11),6),rep(c(5,rep(0,10)),6),rep(rep(0,11),6))
Cmt  <- c(rep(1,6*11),rep(c(2,0,0,0,0,0,0,1,0,0,0),6),rep(2,6*11))
Conc <- rep(0,18*11)

Data <- as.data.frame(list(ID=ID,Time=Time,Dose=Dose,Rate=Rate,Cmt=Cmt,Conc=Conc))

SimData <- groupedData( Conc ~ Time | ID,
  data = Data,
  labels = list( x = "Time", y = "Concentration"))

OneCompAbsModel <- nlmeODE(OneCompAbs,SimData)

kaSim <- rep(log(rep(0.05,18))+0.3*rnorm(18),each=11)
CLSim <- rep(log(rep(0.5,18))+0.2*rnorm(18),each=11)
V1Sim <- rep(log(rep(10,18))+0.1*rnorm(18),each=11)
F1Sim <- rep(log(0.8),18*11)

SimData$Sim <- OneCompAbsModel(kaSim,CLSim,V1Sim,F1Sim,SimData$Time,SimData$ID)

SimData$Conc <- SimData$Sim + 0.3*rnorm(dim(SimData)[1])

Data <- groupedData( Conc ~ Time | ID,
  data = SimData,
  labels = list( x = "Time", y = "Concentration"))

plot(Data,aspect=1/1)

#Estimation of model parameters
OneCompAbsModel <- nlmeODE(OneCompAbs,Data)

## Not run:
fit1 <- nlme(Conc ~ OneCompAbsModel(ka,CL,V1,F1,Time,ID),
  data = Data, fixed=ka+CL+V1+F1~1, random = pdDiag(ka+CL+V1~1),
  start=c(ka=log(0.05),CL=log(0.5),V1=log(10.0),F1=log(0.8)),
  control=list(msVerbose=TRUE,pnlsTol=1),
  verbose=TRUE)

plot(augPred(fit1,level=0:1,length.out=300),aspect=1/1)

## End(Not run)

#Estimation of rate of infusion
Data$Rate[Data$Rate==5] <- -1

OneCompAbs <- list(DiffEq=list(
  dA1dt = ~ -ka*A1,
```

```

        dA2dt = ~ ka*A1 - CL/V1*A2),
  ObsEq=list(
    SC= ~0,
    C = ~ A2/V1),
  States=c("A1", "A2"),
  Parms=c("ka", "CL", "V1", "F1", "Rate"),
  Init=list(0,0))

OneCompAbsModel <- nlmeODE(OneCompAbs,Data)

## Not run:
fit2 <- nlme(Conc ~ OneCompAbsModel(ka,CL,V1,F1,Rate,Time,ID),
  data = Data, fixed=ka+CL+V1+F1+Rate~1, random = pdDiag(ka+CL+V1~1),
  start=c(ka=log(0.05),CL=log(0.5),V1=log(10.0),F1=log(0.8),Rate=log(5)),
  control=list(msVerbose=TRUE,pnlsTol=1),
  verbose=TRUE)

plot(augPred(fit2,level=0:1,length.out=300),aspect=1/1)

## End(Not run)

#Estimation of length of infusion
Data$Rate[Data$Rate==-1] <- -2

OneCompAbs <- list(DiffEq=list(
  dA1dt = ~ -ka*A1,
  dA2dt = ~ ka*A1 - CL/V1*A2),
  ObsEq=list(
    SC= ~0,
    C = ~ A2/V1),
  States=c("A1", "A2"),
  Parms=c("ka", "CL", "V1", "F1", "Tcrit"),
  Init=list(0,0))

OneCompAbsModel <- nlmeODE(OneCompAbs,Data)

## Not run:
fit3 <- nlme(Conc ~ OneCompAbsModel(ka,CL,V1,F1,Tcrit,Time,ID),
  data = Data, fixed=ka+CL+V1+F1+Tcrit~1, random = pdDiag(ka+CL+V1~1),
  start=c(ka=log(0.05),CL=log(0.5),V1=log(10.0),F1=log(0.8),Tcrit=log(20)),
  control=list(msVerbose=TRUE,pnlsTol=1),
  verbose=TRUE)

plot(augPred(fit3,level=0:1,length.out=300),aspect=1/1)

## End(Not run)

#####
### Simulation and simultaneous estimation of PK/PD data ###
#####

PoolModel <- list(
  DiffEq=list(

```

```

      dy1dt = ~ -ke*y1,
      dy2dt = ~ krel * (1-Emax*(y1/Vd)**gamma/(EC50**gamma+(y1/Vd)**gamma)) * y3 - kout * y2,
      dy3dt = ~ Kin - krel * (1-Emax*(y1/Vd)**gamma/(EC50**gamma+(y1/Vd)**gamma))*y3),
      ObsEq=list(
        PK   = ~ y1/Vd,
        PD   = ~ y2,
        Pool = ~ 0),
      States=c("y1","y2","y3"),
      Params=c("ke","Vd","Kin","kout","krel","Emax","EC50","gamma"),
      Init=list(0,"Kin/kout","Kin/krel"))

ID   <- rep(seq(1:12),each=2*12)
Time <- rep(rep(c(0,0.25,0.5,0.75,1,2,4,6,8,10,12,24),each=2),12)
Dose <- rep(c(100,rep(0,23)),12)
Cmt  <- rep(rep(c(1,2),12),12)
Type <- rep(rep(c(1,2),12),12)
Conc <- rep(0,2*12*12)

Data <- as.data.frame(list(ID=ID,Time=Time,Dose=Dose,Cmt=Cmt,Type=Type,Conc=Conc))

SimData <- groupedData( Conc ~ Time | ID/Type,
  data = Data,
  labels = list( x = "Time", y = "Concentration"))

PKPDpoolModel <- nlmeODE(PoolModel,SimData)

keSim   <- rep(log(rep(0.05,12))+0.1*rnorm(12),each=2*12)
VdSim   <- rep(log(rep(10,12))+0.01*rnorm(12),each=2*12)
EC50Sim <- rep(log(rep(5,12))+0.1*rnorm(12),each=2*12)
KinSim  <- rep(log(5),2*12*12)
koutSim <- rep(log(0.5),2*12*12)
krelSim <- rep(log(2),2*12*12)
EmaxSim <- rep(log(1),2*12*12)
gammaSim <- rep(log(3),2*12*12)

SimData$Sim <- PKPDpoolModel(keSim,VdSim,KinSim,koutSim,krelSim,EmaxSim,EC50Sim,
  gammaSim,SimData$Time,SimData$ID,SimData$Type)
SimData$Conc[SimData$Type==1] <- SimData$Sim[SimData$Type==1]*(1 +
  0.1*rnorm(length(SimData[SimData$Type==1,1])))
SimData$Conc[SimData$Type==2] <- SimData$Sim[SimData$Type==2]*(1 +
  0.05*rnorm(length(SimData[SimData$Type==2,1])))

Data <- groupedData( Conc ~ Time | ID/Type,
  data = SimData,
  labels = list( x = "Time", y = "Concentration"))

plot(Data,display=1,aspect=1/1)

#Fixed parameters
Data$Emax <- rep(log(1),dim(Data)[1])

#Estimation of model parameters
PKPDpoolModel <- nlmeODE(PoolModel,Data)

```

```

## Not run:
PKPDpool.nlmf <- nlme(log(Conc) ~ log(PKPDpoolModel(ke,Vd,Kin,kout,krel,Emax,EC50,gamma,Time,ID,Type)),
  data = Data, fixed=ke+Vd+Kin+kout+krel+EC50+gamma~1, random = pdDiag(ke+Vd+EC50~1),
  groups=~ID,
  weights=varIdent(form=~1|Type),
  start=c(ke=log(0.05),Vd=log(10),Kin=log(5),kout=log(0.5),krel=log(2),EC50=log(5),gamma=log(3)),
  control=list(msVerbose=TRUE,msMaxIter=20,pnlsMaxIter=20,pnlsTol=1),
  verbose=TRUE)

PKPDpool.nlmf
exp(fixef(PKPDpool.nlmf))

#Plot results
ni <- 100

TimeSim <- seq(from=0,to=24,length=ni)
TimeSim <- rep(rep(TimeSim,each=2),12)

SubjectSim <- rep(1:12,each=2*ni)
TypeSim <- rep(rep(c(1,2),ni),12)

IndCoef <- coef(PKPDpool.nlmf)
IpredSim <- PKPDpoolModel(
  rep(IndCoef[,1],each=2*ni),
  rep(IndCoef[,2],each=2*ni),
  rep(IndCoef[,3],each=2*ni),
  rep(IndCoef[,4],each=2*ni),
  rep(IndCoef[,5],each=2*ni),
  rep(rep(log(1),12),each=2*ni),
  rep(IndCoef[,6],each=2*ni),
  rep(IndCoef[,7],each=2*ni),
  TimeSim,SubjectSim,TypeSim)

PopCoef <- fixef(PKPDpool.nlmf)
PredSim <- PKPDpoolModel( rep(rep(PopCoef[1],12),each=2*ni),
  rep(rep(PopCoef[2],12),each=2*ni),
  rep(rep(PopCoef[3],12),each=2*ni),
  rep(rep(PopCoef[4],12),each=2*ni),
  rep(rep(PopCoef[5],12),each=2*ni),
  rep(rep(log(1),12),each=2*ni),
  rep(rep(PopCoef[6],12),each=2*ni),
  rep(rep(PopCoef[7],12),each=2*ni),
  TimeSim,SubjectSim,TypeSim)

plotPool <- as.data.frame(rbind(cbind(TimeSim,SubjectSim,PredSim,TypeSim,rep("Pred",2400)),
  cbind(TimeSim,SubjectSim,IpredSim,TypeSim,rep("Ipred",2400)),
  cbind(Data$Time,Data$ID,Data$Conc,Data$Type,rep("Obs",288))
  ))
names(plotPool) <- c("Time","Subject","Conc","Type","Flag")

plotPool$Subject <- as.factor(as.numeric(as.character(plotPool$Subject)))
plotPool$Type <- as.factor(plotPool$Type)

```

```

plotPool$Flag <- as.factor(plotPool$Flag)
plotPool$Conc <- as.numeric(as.character(plotPool$Conc))
plotPool$Time <- as.numeric(as.character(plotPool$Time))

plotPoolPK <- subset(plotPool,Type==1)
plotPoolPD <- subset(plotPool,Type==2)

get(getOption("device"))(record=TRUE,width=9,height=9)
xyplot (Conc~Time | Subject, data=plotPoolPK,
        layout=c(4,3),
        aspect=1/1,
        groups=Flag,
        xlab="Time since drug administration (hr)",
        ylab="PK concentration (ng/mL)",
        key=list(x=0,y=1,corner=c(0,0),transparent=TRUE,
                text = list(c("Population", "Individual","Observed")),
                lines = list(type=c("l","l","p"), pch=16, col=c(1,"red",1),
lty=c(2,1,1)),columns=3),
        strip = function(...) strip.default(..., strip.names=c(FALSE,TRUE), style=1),
        panel = function(x, y, groups,...) {
            panel.grid(h=3,v=3,col="lightgray",lwd=0.7,...)
            panel.superpose.2(x,y,groups,type=c("l","p","l"),
col=c("red",1,1), lty=c(1,1,2),pch=16, lwd=1.4,...)},
        par.strip.text=list(cex=1.0))

xyplot (Conc~Time | Subject, data=plotPoolPD,
        layout=c(4,3),
        aspect=1/1,
        groups=Flag,
        xlab="Time since drug administration (hr)",
        ylab="PD concentration (ng/mL)",
        key=list(x=0,y=1,corner=c(0,0),transparent=TRUE,
                text = list(c("Population", "Individual","Observed")),
                lines = list(type=c("l","l","p"), pch=16, col=c(1,"red",1),
lty=c(2,1,1)),columns=3),
        strip = function(...) strip.default(..., strip.names=c(FALSE,TRUE), style=1),
        panel = function(x, y, groups,...) {
            panel.grid(h=3,v=3,col="lightgray",lwd=0.7,...)
            panel.superpose.2(x,y,groups,type=c("l","p","l"),
col=c("red",1,1), lty=c(1,1,2),pch=16, lwd=1.4,...)},
        par.strip.text=list(cex=1.0))

## End(Not run)

#####
### Minimal Model of Glucose and Insulin      ###
#####

MMmodel <- list(
  DiffEq=list(
    dgdtdt = ~ Sg*Gb - (Sg+x)*g,
    dxdt = ~ -p2*(x-Si*(i-Ib)),

```

```

    didt = ~ -n*(i-Ib)+gamma*(g-h)*t),
  ObsEq=list(
    gc= ~ g,
    xc= ~ 0,
    ic= ~ i),
  States=c("g","x","i"),
  Params=c("Sg","p2","Si","n","gamma","h","Gb","Ib","G0","I0"),
  Init=list("G0",0,"I0")
)

id   <- rep(seq(1:12),each=2*29)
time <- rep(rep(c(0,2,3,4,5,6,8,10,12,14,16,19,22,24,25,27,30,35,40,50,60,70,80,90,100,120,140,160,180),
each=2),12)
type <- rep(rep(c(1,2),29),12)
conc <- rep(0,2*12*29)

data <- as.data.frame(list(id=id,time=time,type=type,conc=conc))

MMData <- groupedData(conc~time|id/type,data=data,
  labels=list(x="Time",y="Concentration"))

Sgsim <- rep(rep(log(0.025),12)+0.2*rnorm(12),each=2*29)
p2sim <- rep(rep(log(0.007),12)+0*rnorm(12),each=2*29)
Sisim <- rep(rep(log(0.001),12)+0.3*rnorm(12),each=2*29)
nsim <- rep(rep(log(0.15),12)+0*rnorm(12),each=2*29)
gammasim <- rep(rep(log(0.001),12)+0*rnorm(12),each=2*29)
hsim <- rep(rep(log(65),12)+0*rnorm(12),each=2*29)
Gbsim <- rep(rep(log(100),12)+0*rnorm(12),each=2*29)
Ibsim <- rep(rep(log(10),12)+0*rnorm(12),each=2*29)

G0sim <- rep(rep(log(250),12)+0.2*rnorm(12),each=2*29)
I0sim <- rep(rep(log(120),12)+0*rnorm(12),each=2*29)

MinModel <- nlmeODE(MMmodel,MMData)

data$sim<-MinModel(Sgsim,p2sim,Sisim,nsim,gammasim,hsim,Gbsim,Ibsim,G0sim,I0sim,
data$time,data$id,data$type)
data$conc[data$type==1] <- data$sim[data$type==1]*(1+0.2*rnorm(length(data[data$type==1,1])))
data$conc[data$type==2] <- data$sim[data$type==2]*(1+0.2*rnorm(length(data[data$type==2,1])))

data$Gb <- Gbsim
data$Ib <- Ibsim
MMData <- groupedData( conc ~ time | id/type,
  data = data,
  labels = list( x = "Time", y = "Concentration"))

plot(MMData,display=1,aspect=1/1)

MinModel <- nlmeODE(MMmodel,MMData)

## Not run:
MM.nlme <-nlme(conc~MinModel(Sg,p2,Si,n,gamma,h,Gb,Ib,G0,I0,time,id,type),
  data=MMData, fixed=Sg+p2+Si+n+gamma+h+G0+I0~1,

```

```

        groups=~id,
        weights=varExp(0.2,form=~fitted(.)|type),
        random=pdDiag(Sg+Si+G0~1),
        start=c(Sg=log(0.025),p2=log(0.007),Si=log(0.001),n=log(0.15),
gamma=log(0.001),h=log(65),G0=log(250),I0=log(120)),
        control=list(returnObject=TRUE,msVerbose=TRUE,msMaxIter=20,pnlMaxIter=20,pnlTol=1),
        verbose=TRUE)

MM.nlmf
exp(fixef(MM.nlmf))

#Plot results
ni <- 100

TimeSim <- seq(from=0,to=180,length=ni)
TimeSim <- rep(rep(TimeSim,each=2),12)

SubjectSim <- rep(1:12,each=2*ni)
TypeSim <- rep(rep(c(1,2),ni),12)

IndCoef <- coef(MM.nlmf)
IpredSim <- MinModel(
  rep(IndCoef[,1],each=2*ni),
  rep(IndCoef[,2],each=2*ni),
  rep(IndCoef[,3],each=2*ni),
  rep(IndCoef[,4],each=2*ni),
  rep(IndCoef[,5],each=2*ni),
  rep(IndCoef[,6],each=2*ni),
  rep(rep(unique(MMData$Gb),12),each=2*ni),
  rep(rep(unique(MMData$Ib),12),each=2*ni),
  rep(IndCoef[,7],each=2*ni),
  rep(IndCoef[,8],each=2*ni),
  TimeSim,SubjectSim,TypeSim)

PopCoef <- fixef(MM.nlmf)
PredSim <- MinModel(
  rep(rep(PopCoef[1],12),each=2*ni),
  rep(rep(PopCoef[2],12),each=2*ni),
  rep(rep(PopCoef[3],12),each=2*ni),
  rep(rep(PopCoef[4],12),each=2*ni),
  rep(rep(PopCoef[5],12),each=2*ni),
  rep(rep(PopCoef[6],12),each=2*ni),
  rep(rep(unique(MMData$Gb),12),each=2*ni),
  rep(rep(unique(MMData$Ib),12),each=2*ni),
  rep(rep(PopCoef[7],12),each=2*ni),
  rep(rep(PopCoef[8],12),each=2*ni),
  TimeSim,SubjectSim,TypeSim)

plotMM <- as.data.frame(rbind(cbind(TimeSim,SubjectSim,PredSim,TypeSim,rep("Pred",2400)),
  cbind(TimeSim,SubjectSim,IpredSim,TypeSim,rep("Ipred",2400)),
  cbind(MMData$time,MMData$id,MMData$conc,MMData$type,rep("Obs",696))
))
names(plotMM) <- c("Time","Subject","Conc","Type","Flag")

```

```

plotMM$Subject <- as.factor(as.numeric(as.character(plotMM$Subject)))
plotMM$Type <- as.factor(plotMM$Type)
plotMM$Flag <- as.factor(plotMM$Flag)
plotMM$Conc <- as.numeric(as.character(plotMM$Conc))
plotMM$Time <- as.numeric(as.character(plotMM$Time))

plotMMG <- subset(plotMM, Type==1)
plotMMI <- subset(plotMM, Type==2)

get(getOption("device"))(record=TRUE,width=9,height=9)
xyplot (Conc~Time | Subject, data=plotMMG,
        layout=c(4,3),
        aspect=1/1,
        groups=Flag,
        xlab="Time (hr)",
        ylab="Glucose concentration",
        key=list(x=0,y=1,corner=c(0,0),transparent=TRUE,
                text = list(c("Population", "Individual", "Observed")),
                lines = list(type=c("l", "l", "p"), lwd=3, pch=16, col=c(1,"red",1),
lty=c(1,1,1)),columns=3),
        strip = function(...) strip.default(..., strip.names=c(FALSE,TRUE), style=1),
        panel = function(x, y, groups,...) {
          panel.abline(h=c(100,200,300,400),col="lightgray",lwd=0.7,...)
          panel.abline(v=c(0,50,100,150,200),col="lightgray",lwd=0.7,...)
          panel.superpose.2(x,y,groups,type=c("l", "p", "l"),
col=c("red",1,1), lty=c(1,1,1),pch=16,cex=.5, lwd=2,...)},
        par.strip.text=list(cex=1.0))

xyplot (Conc~Time | Subject, data=plotMMI,
        layout=c(4,3),
        aspect=1/1,
        groups=Flag,
        xlab="Time (hr)",
        ylab="Insulin concentration",
        key=list(x=0,y=1,corner=c(0,0),transparent=TRUE,
                text = list(c("Population", "Individual", "Observed")),
                lines = list(type=c("l", "l", "p"), lwd=3, pch=16, col=c(1,"red",1),
lty=c(1,1,1)),columns=3),
        strip = function(...) strip.default(..., strip.names=c(FALSE,TRUE), style=1),
        panel = function(x, y, groups,...) {
          panel.abline(h=c(100,200,300,400),col="lightgray",lwd=0.7,...)
          panel.abline(v=c(0,50,100,150,200),col="lightgray",lwd=0.7,...)
          panel.superpose.2(x,y,groups,type=c("l", "p", "l"),
col=c("red",1,1), lty=c(1,1,1),pch=16,cex=.5, lwd=2,...)},
        par.strip.text=list(cex=1.0))

## End(Not run)

#####
### Minimal Model of Glucose using observed insulin as forcing function ###
#####

```

```

idata <- data$conc[data$type==2]
data <- data[data$type==1,]
data$i <- idata

MMmodel <- list(
  DiffEq=list(
    dgdt = ~ Sg*Gb - (Sg+x)*g,
    dxdt = ~ -p2*(x-Si*(Insulin(t,id)-Ib))),
  ObsEq=list(
    gc= ~ g,
    xc= ~ 0),
  States=c("g","x"),
  Params=c("Sg","p2","Si","Gb","Ib","id","G0"),
  Init=list("G0",0)
)

Insulin <- function(t,subject){
  subject <- as.integer(log(subject))
  dT <- MMData$time[MMData$id==subject & MMData$time>t][1]-rev(MMData$time[MMData$id==subject & MMData$time
dInsulin <- MMData$i[MMData$id==subject & MMData$time>t][1]-rev(MMData$i[MMData$id==subject & MMData$time
  if(t>=max(MMData$time[MMData$id==subject])){
    conc <- rev(MMData$i[MMData$id==subject & MMData$time<=t])[1]
  }else{
    conc <- rev(MMData$i[MMData$id==subject & MMData$time<=t])[1] +
dInsulin/dT*(t-rev(MMData$time[MMData$id==subject & MMData$time<=t])[1])
  }
  names(conc) <- NULL
  return(conc)
}

MMData <- groupedData( conc ~ time | id,
  data = data,
  labels = list( x = "Time", y = "Concentration"))

plot(MMData,aspect=1/1)

MinModel <- nlmeODE(MMmodel,MMData)

## Not run:
MMglucose.nlme <-nlme(log(conc)~log(MinModel(Sg,p2,Si,Gb,Ib,id,G0,time,id)),
  data=MMData, fixed=Sg+p2+Si+G0~1,
  random=pdDiag(Sg+Si+G0~1),
  start=c(Sg=log(0.03),p2=log(0.01),Si=log(0.001),G0=log(250)),
  control=list(returnObject=TRUE,msVerbose=TRUE,msMaxIter=20,pnlMaxIter=20,pnlTol=1),
  verbose=TRUE)

MMglucose.nlme
exp(fixef(MMglucose.nlme))

#Plot results
ni <- 100

TimeSim <- seq(from=0,to=180,length=ni)
TimeSim <- rep(rep(TimeSim,each=2),12)

```

```

SubjectSim <- rep(1:12,each=2*ni)

IndCoef <- coef(MMglucose.nlm)
IpredSim <- MinModel(
  rep(IndCoef[,1],each=2*ni),
  rep(IndCoef[,2],each=2*ni),
  rep(IndCoef[,3],each=2*ni),
  rep(rep(unique(MMData$Gb),12),each=2*ni),
  rep(rep(unique(MMData$Ib),12),each=2*ni),
  SubjectSim,
  rep(IndCoef[,4],each=2*ni),
  TimeSim,SubjectSim)

PopCoef <- fixef(MMglucose.nlm)
PredSim <- MinModel( rep(rep(PopCoef[1],12),each=2*ni),
  rep(rep(PopCoef[2],12),each=2*ni),
  rep(rep(PopCoef[3],12),each=2*ni),
  rep(rep(unique(MMData$Gb),12),each=2*ni),
  rep(rep(unique(MMData$Ib),12),each=2*ni),
  SubjectSim,
  rep(rep(PopCoef[4],12),each=2*ni),
  TimeSim,SubjectSim)

plotMM <- as.data.frame(rbind(cbind(TimeSim,SubjectSim,PredSim,rep("Pred",2400)),
  cbind(TimeSim,SubjectSim,IpredSim,rep("Ipred",2400)),
  cbind(MMData$time,MMData$id,MMData$conc,rep("Obs",348))
))
names(plotMM) <- c("Time","Subject","Conc","Flag")

plotMM$Subject <- as.factor(as.numeric(as.character(plotMM$Subject)))
plotMM$Flag <- as.factor(plotMM$Flag)
plotMM$Conc <- as.numeric(as.character(plotMM$Conc))
plotMM$Time <- as.numeric(as.character(plotMM$Time))

xyplot (Conc~Time | Subject, data=plotMMG,
  layout=c(4,3),
  aspect=1/1,
  groups=Flag,
  xlab="Time (hr)",
  ylab="Glucose concentration",
  key=list(x=0,y=1,corner=c(0,0),transparent=TRUE,
  text = list(c("Population", "Individual","Observed"))),
  lines = list(type=c("l","l","p"), pch=16, lwd=3, col=c(1,"red",1),
lty=c(1,1,1)),columns=3),
  strip = function(...) strip.default(..., strip.names=c(FALSE,TRUE), style=1),
  panel = function(x, y, groups,...) {
    panel.grid(h=3,v=3,col="lightgray",lwd=0.7,...)
    panel.superpose.2(x,y,groups,type=c("l","p","l"),
col=c("red",1,1), lty=c(1,1,1),pch=16, cex=.5, lwd=3,...)},
  par.strip.text=list(cex=1.0))

```

```
## End(Not run)
```

Index

*Topic **models**

nlmeODE, 2

PKPDmodels, 3

lsoda, 3

nlme, 3

nlmeODE, 2

PKPDmodels, 3, 3