A Switching Markov Chain Monte Carlo Method for Statistical Identifiability of Nonlinear Pharmacokinetics Models

Seongho Kim* and Lang Li*

University of Louisville and Indiana University School of Medicine

Supplementary Material

The R codes are provided for single component MCMC (SCM), group component MCMC (GCM), and switching MCMC (SWM) algorithms.

```
# required R packages
library(mnormt)
library(numDeriv)
library(nlmeODE)
library(pscl)
library(odesolve)

# Mid IV two-compartment kinetic model
Mid.kinetic.ODE <- list(
  DiffEq = list(
    dy1dt =~ -80*(Vmax/(Km+y1/V1))/(80+(Vmax/(Km+y1/V1)))*y1/V1+CL12*(-y1/V1+y2/V2),
    dy2dt =~ CL12*(y1/V1-y2/V2)
  ),
  ObsEq = list(
    c1 = ~ y1/V1,
    c2 = ~ 0
  ),
  States = c("y1","y2"),
  Parms = c("V1","V2","Vmax","Km","CL12"),
  Init=list(0,0)
)

# calculate the concentration using Mid.kinetic.ODE on "nlmeODE"
cal.conc <- function(p,data,MODEL){
  len = length(data$Time)
  MidM = nlmeODE(MODEL,data)
  V1Sim = rep(p[1],len)
```

```
  V2Sim = rep(p[2],len)
  VmaxSim = rep(p[3],len)
  KmSim = rep(p[4],len)
  CL12Sim = rep(p[5],len)
  new.conc = MidM(V1Sim,V2Sim,VmaxSim,KmSim,CL12Sim,data$Time,data$ID)
  new.conc
}

# first derivative
f.der <- function(para,data){
  fn0 = function(e.par){
    e.v1 = exp(e.par[1])
    e.pred = cal.conc(p=e.par,data=data,MODEL=Mid.kinetic.ODE)
    t.pred = e.pred[,2]/e.v1
    t.pred = cbind(t.pred,conc0)
    t.pred = na.omit(t.pred)
    pred = log(t.pred[,1])
    pred
  }
  der = jacobian(fn0,para)
  der
}

# Fisher information matrix
f.fisher <- function(para,data,sigma2){
  der.f = f.der(para=para,data=data)
  fisher.f = (t(der.f) %*% der.f)/sigma2
  fisher.f
}

# proposal or kernel function for MCMC
proposal <- function(theta,var){
  as.vector(rmnorm(1,theta,var))
}

# Gibbs for sigma2
sigma2.Gibbs <- function(data,para,prior.sigma2){
  v10 = exp(para[1])
  conc0 = data$Conc
  a0 = prior.sigma2[1]
  b0 = prior.sigma2[2]
  ll = 0
  n.pred = cal.conc(p=para,data=data,MODEL=Mid.kinetic.ODE)
  n.conc = (n.pred[,2]/v10)
  n.data = na.omit(cbind(conc0,n.conc))
  n.data = n.data[n.data[,2]>0,]
```

```
  n.data[,2] = log(n.data[,2])
  for(i in 1:dim(n.data)[1]){
    ll = ll + ( ( (n.data[i,1])-(n.data[i,2]) )^2 )/2
  }
  alpha = a0+dim(n.data)[1]/2
  beta = ll + b0
  n.sigma2 = rigamma(1,alpha,beta)
  n.sigma2
}

# log-likelihood
logll <- function(data,sub.para,para,sigma2,prior.mean,prior.cov){
  v10 = exp(para[1])
  conc0 = data$Conc
  ll = 0
  n.pred = cal.conc(p=para,data=data,MODEL=Mid.kinetic.ODE)
  n.conc = (n.pred[,2]/v10)
  n.data = na.omit(cbind(conc0,n.conc))
  n.data = n.data[n.data[,2]>0,]
  n.data[,2] = log(n.data[,2])
  for(i in 1:dim(n.data)[1]){
    ll = ll + (1/sigma2)*( ( (n.data[i,1])-(n.data[i,2]) )^2 )/2
  }
  ll = ll + .5*t(para-prior.mean) %*% solve(prior.cov) %*% (para-prior.mean)
  -ll
}

# log-likelihood for joint
logll.joint <- function(data,para,sigma2,prior.mean,prior.cov){
  v10 = exp(para[1])
  conc0 = data$Conc
  ll = 0
  n.pred = cal.conc(p=para,data=data,MODEL=Mid.kinetic.ODE)
  n.conc = (n.pred[,2]/v10)
  n.data = na.omit(cbind(conc0,n.conc))
  n.data = n.data[n.data[,2]>0,]
  n.data[,2] = log(n.data[,2])
  for(i in 1:dim(n.data)[1]){
    ll = ll + (1/sigma2)*( ( (n.data[i,1])-(n.data[i,2]) )^2 )/2
  }
  ll = ll + .5*t(para-prior.mean) %*% solve(prior.cov) %*% (para-prior.mean)
  -ll
}

# MCMC for SCM with parameters which may be identifiable
SCM.id.MCMC <- function(data,id.para,nid.para,sigma2,cov,prior.mean,prior.cov,c.opt){
```

```
  new.id.para = c(proposal(theta=id.para,var=c.opt*cov))
  U = runif(1,0,1)
  logll.ii = logll(data=data,sub.para=new.id.para,para=c(new.id.para,nid.para)
                   ,sigma2=sigma2,prior.mean=prior.mean,prior.cov=prior.cov)
  logll.i = logll(data=data,sub.para=id.para,para=c(id.para,nid.para),sigma2=sigma2
                  ,prior.mean=prior.mean,prior.cov=prior.cov)
  arate = min( 1, exp(logll.ii - logll.i) )
  if(U<=arate){
    return(new.id.para)
  }else{
    return(id.para)
  }
}

# MCMC for SCM with parameters which may be unidentifiable
SCM.nid.MCMC <- function(data,id.para,nid.para,sigma2,cov,prior.mean,prior.cov,c.opt){
  new.nid.para = c(proposal(theta=nid.para,var=c.opt*cov))
  U = runif(1,0,1)
  logll.ii = logll(data=data,sub.para=new.nid.para,para=c(id.para,new.nid.para)
                   ,sigma2=sigma2,prior.mean=prior.mean,prior.cov=prior.cov)
  logll.i = logll(data=data,sub.para=nid.para,para=c(id.para,nid.para),sigma2=sigma2
                  ,prior.mean=prior.mean,prior.cov=prior.cov)
  arate = min( 1, exp(logll.ii - logll.i) )
  if(U<=arate){
    return(new.nid.para)
  }else{
    return(nid.para)
  }
}

# MCMC for GCM
GCM.MCMC <- function(data,para,sigma2,cov,prior.mean,prior.cov,c.opt){
  new.para = c(proposal(theta=para,var=c.opt*cov))
  U = runif(1,0,1)
  logll.ii = logll.joint(data=data,para=new.para,sigma2=sigma2
                         ,prior.mean=prior.mean,prior.cov=prior.cov)
  logll.i = logll.joint(data=data,para=para,sigma2=sigma2
                        ,prior.mean=prior.mean,prior.cov=prior.cov)
  arate = min( 1, exp(logll.ii - logll.i) )
  if(U<=arate){
    return(new.para)
  }else{
    return(para)
  }
}
```

```
# Switching MCMC for SCM, GCM, and SWM
Switching.MCMC <- function(case,data,run){
  c.opt = 1,
  sec.pos = 4,
  cut.off = .5
  data0 = data
  conc0 = data0$Conc
  time0 = data0$Time
  para.mle = log(c(8.71,36.86,377.86,2.11,24.6))
  sigma2.mle = .011
  prior.mean = log(c(78,53,4890,2,43))
  prior.cov = diag(.5,5)
  prior.sigma2 = c(2,.011)
  sigma2.prior0 = prior.sigma2
  para.mle0 = para.mle
  para0 = para.mle0
  num.para0 = length(para0)
  id.pos0 = c(1:(sec.pos-1))
  nid.pos0 = c(sec.pos:num.para0)
  id.para0 = para0[id.pos0]
  nid.para0 = para0[nid.pos0]
  sigma20 = sigma2.mle
  fm = f.fisher(para=para.mle,data=data0,sigma2=sigma2.mle)
  id.fm0 = fm[id.pos0,id.pos0]
  nid.fm0 = fm[nid.pos0,nid.pos0]
  id.cov0 = solve(id.fm0)
  nid.cov0 = solve(nid.fm0)
  cov0 = solve(fm)
  output = matrix(0,run,7)
  if(case == 1){
    sflag = 0
  }else if(case == 2){
    sflag = 1
  }else if(case == 3){
    sflag = 1
    cutoff = cut.off
  }
  for(i in 1:run){
    if(sflag==0){
      id.para0 = SCM.id.MCMC(data=data0,id.para=id.para0,nid.para=nid.para0,sigma2=sigma20
                             ,cov=id.cov0,prior.mean=prior.mean,prior.cov=prior.cov
                             ,c.opt=c.opt)
      nid.para0 = SCM.nid.MCMC(data=data0,id.para=id.para0,nid.para=nid.para0,sigma2=sigma20
                               ,cov=nid.cov0,prior.mean=prior.mean
                               ,prior.cov=prior.cov,c.opt=c.opt)
      para0 = c(id.para0,nid.para0)
```

```
    }else{
      para0 = GCM.MCMC(data=data0,para=para0,sigma2=sigma20,cov=cov0,prior.mean=prior.mean
                       ,prior.cov=prior.cov,c.opt=c.opt)
      id.para0 = para0[id.pos0]
      nid.para0 = para0[nid.pos0]
    }
    sigma20 = sigma2.Gibbs(data=data0,para=para0,prior.sigma2=sigma2.prior0)
    output[i,] = c(para0,sigma20,sflag)
    if(case == 3){
      ediff = runif(1)
      if(ediff>= cutoff){
        sflag = 1
      }else{
        sflag = 0
      }
    }
  }
  output
}

# Run all the MCMCs
RUN.SMCMC <- function(data,numrun = 10000){
  # The data must follow the format which is decribed
  # in the R package "nlmeODE."

  # SCM
  scg = Switching.MCMC(case = 1,data = data,run=numrun)

  # GCM
  gcg = Switching.MCMC(case = 2,data = data,run=numrun)

  # SWM
  swcg = Switching.MCMC(case = 3,data = data,run=numrun)

  # OUTPUT
  cbind(scg,gcg,swcg)
}
```