Saemix fits with different error models

Emmanuelle

30/04/2020

Saemix fits with different error models

Objective

Investigate why sometimes saemix doesn't converge and leads to outlandish estimates including variabilities of several thousands.

Simulation using the same setting as Dubois et al. 2011

```
# Parameters
psi1<-c(1.5, 5, 0.04)
omega1<-diag(c(0.05, 0.0125, 0.05))
res1<-c(0.1,0.1)

# Model
model1cpt<-function(psi,id,xidep) {
  tim<-xidep[,1]
  dose<-xidep[,2]
  ka<-psi[id,1]
  V<-psi[id,2]
  CL<-psi[id,3]
  k<-CL/V
  ypred<-dose*ka/(V*(ka-k))*(exp(-k*tim)-exp(-ka*tim))
  return(ypred)
}

# Settings
N <- 50
tim <- c(0,0.25,0.5,1,2,3.5,5,7,9,12,24)
dose<-4

simdat<-data.frame(id=rep(1:N,each=length(tim)),time=rep(tim,N),dose=dose)

psipar<-do.call(rbind,rep(list(psi1),N))
for(i in 1:3) psipar[,i]<-psipar[,i]*exp(rnorm(N,mean=0,sd=sqrt(omega1[i,i])))
summary(psipar)
```

```
##        V1               V2              V3
##  Min.   :0.9728   Min.   :4.322   Min.   :0.02303
##  1st Qu.:1.2821   1st Qu.:4.830   1st Qu.:0.03595
##  Median :1.4315   Median :5.044   Median :0.03951
```

```
##  Mean   :1.4987   Mean   :5.246   Mean   :0.04160
##  3rd Qu.:1.6741   3rd Qu.:5.735   3rd Qu.:0.04562
##  Max.   :3.0752   Max.   :6.898   Max.   :0.07691
```

```
apply(psipar,2,sd)
```

```
## [1] 0.33218367 0.58641770 0.01025446
```

```
ypred<-model1cpt(psipar,id=1:N,xidep=simdat[,2:3])
gpred<-error(ypred,res1,etype=rep(1,length(ypred)))
simdat$conc<-ypred+rnorm(length(ypred),mean=0,sd=gpred)

# Saemix data
saemix.data<-saemixData(name.data=simdat,header=TRUE,sep="",na=NA, name.group=c("id"), name.predictors=c
```

```
## Using the object called simdat in this R session as the data.
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset simdat
##     Structured data: conc ~ time + dose | id
##     X variable for graphs: time ()
```

```
simdat2<-simdat
simdat2$conc[simdat2$time==0]<-0
saemix.data2<-saemixData(name.data=simdat2,header=TRUE,sep="",na=NA, name.group=c("id"), name.predictors
```

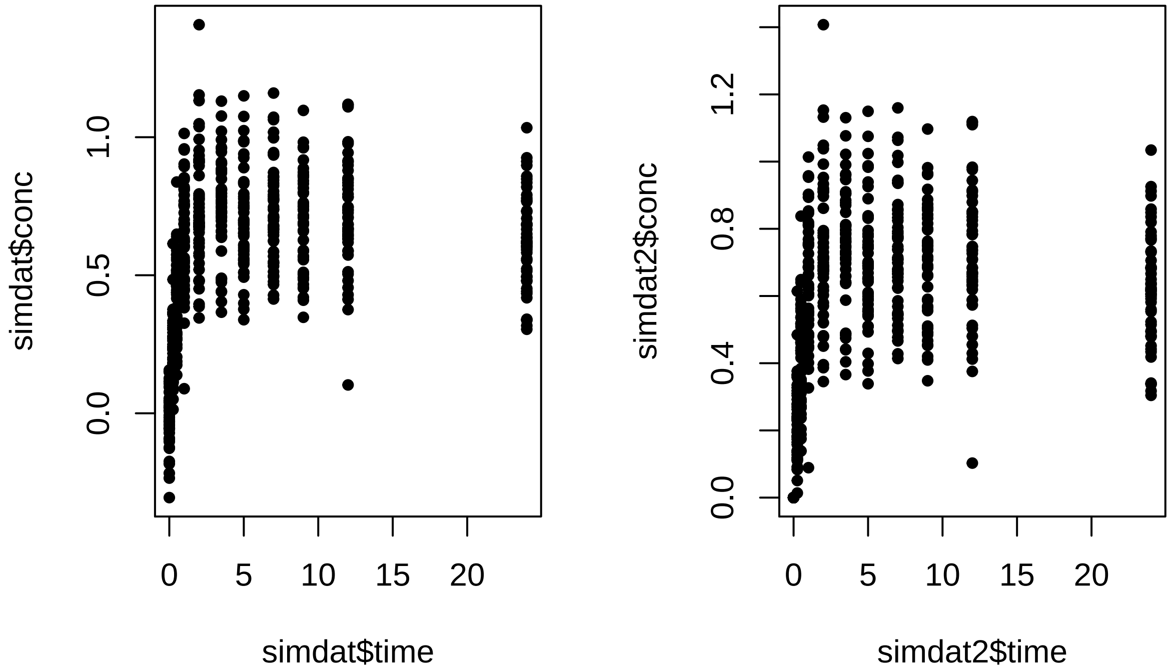
```
## Using the object called simdat2 in this R session as the data.
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset simdat2
##     Structured data: conc ~ time + dose | id
##     X variable for graphs: time ()
```

```
simdat3<-simdat2[simdat2$time>0,]
saemix.data3<-saemixData(name.data=simdat3,header=TRUE,sep="",na=NA, name.group=c("id"), name.predictors
```

```
## Using the object called simdat3 in this R session as the data.
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset simdat3
##     Structured data: conc ~ time + dose | id
##     X variable for graphs: time ()
```

```
par(mfrow=c(1,2))
plot(simdat$time,simdat$conc,pch=20)
```

```r
plot(simdat2$time,simdat2$conc,pch=20)
```




```r
if(FALSE) {
  namfile<-file.path(melDir,"simulationsTlag.csv")
  simdat.mlx<-simdat
  simdat.mlx$dose[simdat.mlx$time>0]<-NA
  write.table(simdat.mlx,namfile, row.names=FALSE, quote=FALSE,na=".")
}
```

Showing results:

```r
print(fit1@results)
```

```
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate SE     CV(%)
## [1,] ka        1.598    0.1149  7.2
## [2,] V         5.264    0.1106  2.1
## [3,] CL        0.034    0.0097 28.2
## [4,] a.1       0.164    0.0055  3.4
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate SE     CV(%)
## ka omega2.ka 0.0346   0.0392 113
## V  omega2.V  0.0026   0.0023  89
## CL omega2.CL 0.4182   0.4456 107
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.ka omega2.V omega2.CL
## omega2.ka 1         0        0
```

```
## omega2.V  0         1        0
## omega2.CL 0         0        1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= -397.2913
##       AIC = -383.2913
##       BIC = -369.9071
##
## Likelihood computed by importance sampling
##       -2LL= -395.9822
##       AIC = -381.9822
##       BIC = -368.598
## ----------------------------------------------------
```

```r
print(fit2@results)
```

```
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate SE  CV(%)
## [1,] ka        1.49     NaN NaN
## [2,] V         5.19     NaN NaN
## [3,] CL        0.04     NaN NaN
## [4,] b.1       0.71     NaN NaN
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate SE  CV(%)
## ka omega2.ka 0.25     NaN NaN
## V  omega2.V  0.24     NaN NaN
## CL omega2.CL 0.27     NaN NaN
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.ka omega2.V omega2.CL
## omega2.ka 1         0        0
## omega2.V  0         1        0
## omega2.CL 0         0        1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= NaN
##       AIC = NaN
##       BIC = NaN
##
## Likelihood computed by importance sampling
##       -2LL= 70839.64
##       AIC = 70853.64
##       BIC = 70867.03
## ----------------------------------------------------
```

```r
print(fit2b@results)
```

```
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate SE  CV(%)
## [1,] ka        1.608    NaN NaN
## [2,] V         5.271    NaN NaN
## [3,] CL        0.035    NaN NaN
## [4,] b.1       0.262    NaN NaN
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate SE  CV(%)
## ka omega2.ka 0.0826   NaN NaN
## V  omega2.V  0.0016   NaN NaN
## CL omega2.CL 0.3343   NaN NaN
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.ka omega2.V omega2.CL
## omega2.ka 1         0        0
## omega2.V  0         1        0
## omega2.CL 0         0        1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= NaN
##       AIC = NaN
##       BIC = NaN
##
## Likelihood computed by importance sampling
##       -2LL= NaN
##       AIC = NaN
##       BIC = NaN
## ----------------------------------------------------
```

```r
print(fit3@results)
```

```
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate SE     CV(%)
## [1,] ka        1.554    0.1045  6.7
## [2,] V         5.235    0.1189  2.3
## [3,] CL        0.037    0.0102 27.6
## [4,] a.1       0.094    0.0085  9.0
## [5,] b.1       0.119    0.0160 13.5
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate SE     CV(%)
## ka omega2.ka 0.0371   0.0326  88
```

```
## V  omega2.V  0.0023   0.0025 109
## CL omega2.CL 0.3290   0.4215 128
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.ka omega2.V omega2.CL
## omega2.ka 1         0        0
## omega2.V  0         1        0
## omega2.CL 0         0        1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= -436.5174
##       AIC = -420.5174
##       BIC = -405.2213
##
## Likelihood computed by importance sampling
##       -2LL= -434.7365
##       AIC = -418.7365
##       BIC = -403.4403
## ----------------------------------------------------
```

```r
cat("Variances seem underestimated for V for constant and combined error models, and with modified data
```

```
## Variances seem underestimated for V for constant and combined error models, and with modified data, a
```

```r
print(fit1@results@omega)
```

```
##            ka           V        CL
## ka 0.03462404 0.000000000 0.0000000
## V  0.00000000 0.002575354 0.0000000
## CL 0.00000000 0.000000000 0.4182175
```

```r
print(fit3@results@omega)
```

```
##            ka          V        CL
## ka 0.03713961 0.00000000 0.0000000
## V  0.00000000 0.00232619 0.0000000
## CL 0.00000000 0.00000000 0.3289533
```

```r
print(fit2b@results@omega)
```

```
##            ka           V        CL
## ka 0.08262704 0.000000000 0.0000000
## V  0.00000000 0.001561481 0.0000000
## CL 0.00000000 0.000000000 0.3343098
```

```r
print(fit2c@results@omega)
```

```
##            ka           V        CL
## ka 0.08072211 0.000000000 0.0000000
## V  0.00000000 0.001117933 0.0000000
## CL 0.00000000 0.000000000 0.3042206
```

```r
print(fit2@results@omega)
```

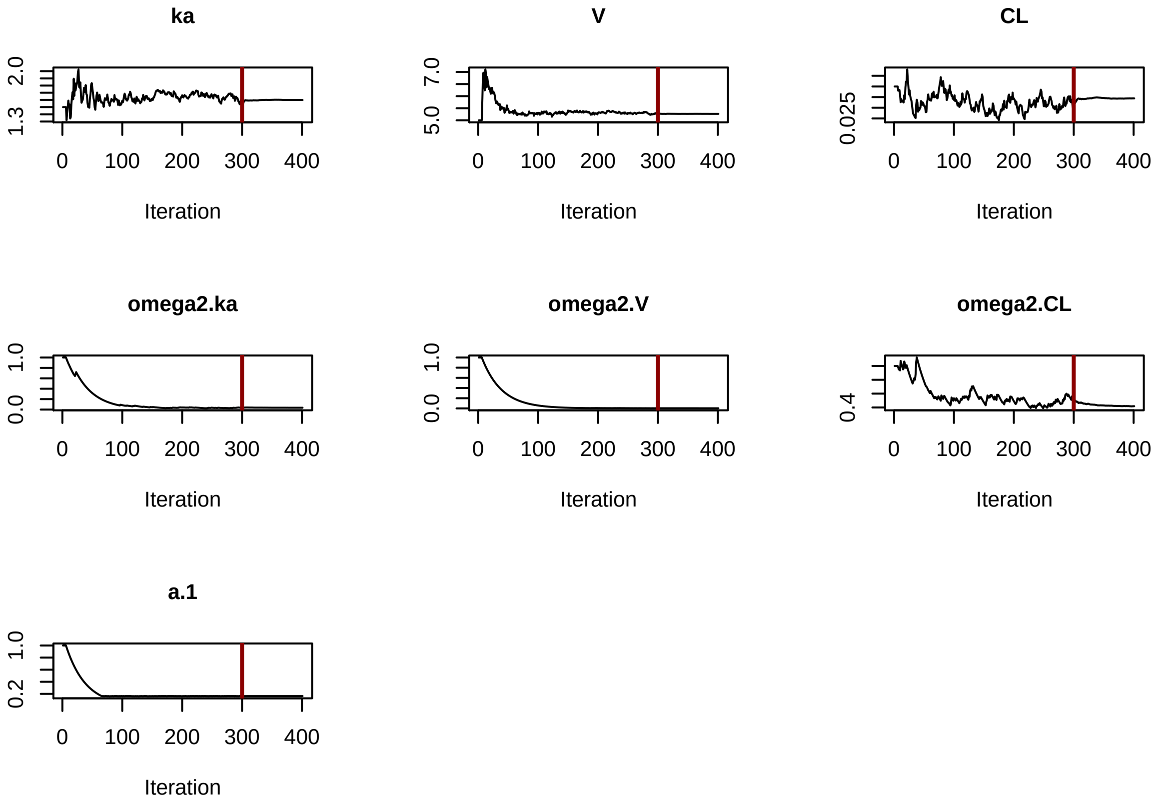
```
##           ka         V        CL
```

```
## ka 0.2453313 0.0000000 0.0000000
## V  0.0000000 0.2372509 0.0000000
## CL 0.0000000 0.0000000 0.2730161
```

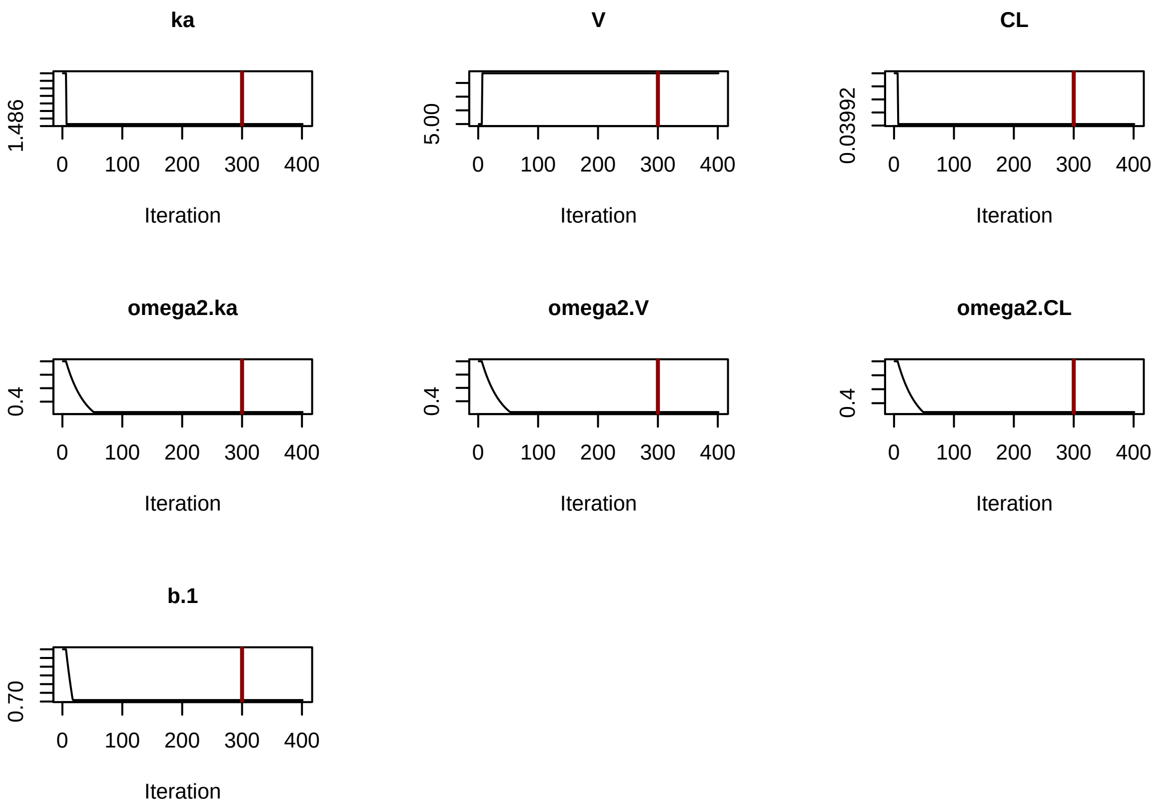
```r
plot(fit1,plot.type="convergence")
```

```
## Plotting convergence plots
```

```r
plot(fit2,plot.type="convergence")
```




```
## Plotting convergence plots
```



Estimation with a proportional error model in Monolix

The runs fail to converge with a proportional error model.

- Model ka, V, Cl

Population parameter estimates VALUE STOCH. APPROX. S.E. R.S.E.(%) Fixed Effects ka_pop 2.87e+64 1.79e+67 6.23e+4 V_pop 2.58e-171 nan nan Cl_pop 1 nan nan Standard Deviation of the Random Effects omega_ka 338 179 53 omega_V 204 27 13.3 omega_Cl 3.6e+56 nan nan Error Model Parameters b 0.3 nan nan

- Model ka, V, Cl, Tlag

Population parameter estimates VALUE STOCH. APPROX. S.E. R.S.E.(%) Fixed Effects Tlag_pop 1 nan nan ka_pop 1.08e+62 2.1e+65 1.94e+5 V_pop 1.34e-171 nan nan Cl_pop 1 nan nan Standard Deviation of the Random Effects omega_Tlag 5.31e+57 nan nan omega_ka 346 538 155 omega_V 207 37.7 18.3 omega_Cl 1.63e+58 nan nan Error Model Parameters b 0.3 nan nan

Problem with the error model, debugging step by step

Initialisation of the algorithm:

```
saemix.model2<-saemixModel(model=model1cpt,description="One compartment model", modeltype="structural",
```

```
##
##
## The following SaemixModel object was successfully created:
##
```

```
## Nonlinear mixed-effects model
##   Model function:  One compartment model  Model type:  structural
## function(psi,id,xidep) {
##   tim<-xidep[,1]
##   dose<-xidep[,2]
##   ka<-psi[id,1]
##   V<-psi[id,2]
##   CL<-psi[id,3]
##   k<-CL/V
##   ypred<-dose*ka/(V*(ka-k))*(exp(-k*tim)-exp(-ka*tim))
##   return(ypred)
## }
## <bytecode: 0x55d9d3737e60>
##   Nb of parameters: 3
##       parameter names:  ka V CL
##       distribution:
##      Parameter Distribution Estimated
## [1,] ka        log-normal   Estimated
## [2,] V         log-normal   Estimated
## [3,] CL        log-normal   Estimated
##   Variance-covariance matrix:
##    ka V CL
## ka  1 0  0
## V   0 1  0
## CL  0 0  1
##   Error model: proportional , initial values: b.1=1
##     No covariate in the model.
##     Initial values
##               ka V   CL
## Pop.CondInit 1.5 5 0.04
## Cov.CondInit 0.0 0 0.00
```

```
saemix.options<-list(nb.chains=3,seed=123456,save=FALSE, save.graphs=FALSE)
saemix.data<-saemixData(name.data=simdat,header=TRUE,sep="",na=NA, name.group=c("id"), name.predictors=
```

```
## Using the object called simdat in this R session as the data.
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset simdat
##     Structured data: conc ~ time + dose | id
##     X variable for graphs: time ()
```

```
  saemixObject<-new(Class="SaemixObject",data=saemix.data,model=saemix.model2,options=saemix.options)
  opt.warn<-getOption("warn")
  if(!saemixObject["options"]$warnings) options(warn=-1)

  saemix.options<-saemixObject["options"]
  saemix.model<-saemixObject["model"]
  saemix.data<-saemixObject["data"]
  saemix.data@ocov<-saemix.data@ocov[saemix.data@data[,"mdv"]==0,,drop=FALSE]
  saemix.data@data<-saemix.data@data[saemix.data@data[,"mdv"]==0,]
```

```r
  saemix.data@ntot.obs<-dim(saemix.data@data)[1]

# Initialisation
  OLDRAND<-TRUE
  set.seed(saemix.options$seed)
  xinit<-initialiseMainAlgo(saemix.data,saemix.model,saemix.options)
  saemix.model<-xinit$saemix.model
  Dargs<-xinit$Dargs
  Uargs<-xinit$Uargs
  varList<-xinit$varList
  phiM<-xinit$phiM
  mean.phi<-xinit$mean.phi
  DYF<-xinit$DYF
  opt<-xinit$opt
  betas<-betas.ini<-xinit$betas
  fixed.psi<-xinit$fixedpsi.ini
  var.eta<-varList$diag.omega

  if (Dargs$modeltype=="structural"){
    theta0<-c(fixed.psi,var.eta[Uargs$i1.omega2],varList$pres[Uargs$ind.res])
    parpop<-matrix(data=0,nrow=(saemix.options$nbiter.tot+1),ncol=(Uargs$nb.parameters+length(Uargs$i1.
    colnames(parpop)<-c(saemix.model["name.modpar"], saemix.model["name.random"], saemix.model["name.sig
    allpar<-matrix(data=0,nrow=(saemix.options$nbiter.tot+1), ncol=(Uargs$nb.betas+length(Uargs$i1.omega
    colnames(allpar)<-c(saemix.model["name.fixed"],saemix.model["name.random"], saemix.model["name.sigma
  } else{
    theta0<-c(fixed.psi,var.eta[Uargs$i1.omega2])
    parpop<-matrix(data=0,nrow=(saemix.options$nbiter.tot+1),ncol=(Uargs$nb.parameters+length(Uargs$i1.
    colnames(parpop)<-c(saemix.model["name.modpar"], saemix.model["name.random"])
    allpar<-matrix(data=0,nrow=(saemix.options$nbiter.tot+1), ncol=(Uargs$nb.betas+length(Uargs$i1.omega
    colnames(allpar)<-c(saemix.model["name.fixed"],saemix.model["name.random"])
  }

  parpop[1,]<-theta0
  allpar[1,]<-xinit$allpar0

  # using several Markov chains - only useful if passed back to main routine...
  #     chdat<-new(Class="SaemixRepData",data=saemix.data, nb.chains=saemix.options$nb.chains)
  #     NM<-chdat["NM"]
  #     IdM<-chdat["dataM"]$IdM
  #     yM<-chdat["dataM"]$yM
  #     XM<-chdat["dataM"][,saemix.data["name.predictors"],drop=FALSE]

# List of sufficient statistics - change during call to stochasticApprox
  suffStat<-list(statphi1=0,statphi2=0,statphi3=0,statrese=0)
  phi<-array(data=0,dim=c(Dargs$N, Uargs$nb.parameters, saemix.options$nb.chains))

# structural model, check nb of parameters
  structural.model<-saemix.model["model"]
  #  nb.parameters<-saemix.model["nb.parameters"]
```

Burn-in iterations:

```r
for (kiter in 1:saemix.options$nbiter.burn) { # Iterative portion of algorithm
# Burn-in - first loop useless
```

```r
# E-step
  xmcmc<-estep(kiter, Uargs, Dargs, opt, structural.model, mean.phi, varList, DYF, phiM)
  varList<-xmcmc$varList
  DYF<-xmcmc$DYF
  phiM<-xmcmc$phiM

  # no M-step during burn-in phase
  allpar[(kiter+1),]<-allpar[kiter,]
  if(Dargs$modeltype=="structural") {
    theta<-c(fixed.psi,var.eta[Uargs$i1.omega2],varList$pres[Uargs$ind.res])
  } else{
    theta<-c(fixed.psi,var.eta[Uargs$i1.omega2])
  }
  parpop[(kiter+1),]<-theta
}
print(theta)
```

```
##   ka    V   CL   ka    V   CL
## 1.50 5.00 0.04 1.00 1.00 1.00 1.00
```

```r
print(head(phiM))
```

```
##           [,1]     [,2]      [,3]
## [1,] 0.8223317 1.738545 -2.643260
## [2,] 0.2674412 2.091693 -2.881193
## [3,] 0.2279642 1.363799 -2.611791
## [4,] 0.4492088 1.650257 -3.580531
## [5,] 1.5315930 1.528553 -3.719117
## [6,] 0.8226952 2.245846 -3.280672
```

```r
print(varList$pres)
```

```
## [1] 0 1
```

Something goes wrong in xstoch:

- sum(Dargs$yobs - fk) * *2/cutoff(fk * *2, .Machine$double.eps) explodes when predictions are very small (fk=0 or smaller than machine precision)
- changed code below to ignore the terms for which fk is too small in the summation

```r
kiter<-saemix.options$nbiter.burn+1
    # E-step
  xmcmc<-estep(kiter, Uargs, Dargs, opt, structural.model, mean.phi, varList, DYF, phiM)
  varList<-xmcmc$varList
  DYF<-xmcmc$DYF
  phiM<-xmcmc$phiM

  # M-step and stochastic Approximation

#  xstoch<-mstep(kiter, Uargs, Dargs, opt, structural.model, DYF, phiM, varList, phi, betas, suffStat)

    # Update variances - TODO - check if here or elsewhere
    nb.etas<-length(varList$ind.eta)
    domega<-cutoff(mydiag(varList$omega[varList$ind.eta,varList$ind.eta]),.Machine$double.eps)
    omega.eta<-varList$omega[varList$ind.eta,varList$ind.eta,drop=FALSE]
    omega.eta<-omega.eta-mydiag(mydiag(varList$omega[varList$ind.eta,varList$ind.eta]))+mydiag(domega)
    #  print(varList$omega.eta)
```

```r
    chol.omega<-try(chol(omega.eta))
    d1.omega<-Uargs$LCOV[,varList$ind.eta]%*%solve(omega.eta)
    d2.omega<-d1.omega%*%t(Uargs$LCOV[,varList$ind.eta])
    comega<-Uargs$COV2*d2.omega

    psiM<-transphi(phiM,Dargs$transform.par)
    fpred<-structural.model(psiM, Dargs$IdM, Dargs$XM)
    for(ityp in Dargs$etype.exp) fpred[Dargs$XM$ytype==ityp]<-log(cutoff(fpred[Dargs$XM$ytype==ityp]))
#   if(Dargs$error.model=="exponential")
#       fpred<-log(cutoff(fpred))
    ff<-matrix(fpred,nrow=Dargs$nobs,ncol=Uargs$nchains)
    for(k in 1:Uargs$nchains) phi[,,k]<-phiM[((k-1)*Dargs$N+1):(k*Dargs$N),]
    # overall speed similar
    #    phi<-aperm(array(phiM,c(N,nchains,3)),c(1,3,2))
    stat1<-apply(phi[,varList$ind.eta,,drop=FALSE],c(1,2),sum) # sum on columns ind.eta of phi, across
    stat2<-matrix(data=0,nrow=nb.etas,ncol=nb.etas)
    stat3<-apply(phi**2,c(1,2),sum) #  sum on phi**2, across 3rd dimension
    statr<-0
    for(k in 1:Uargs$nchains) {
        phik<-phi[,varList$ind.eta,k]
        stat2<-stat2+t(phik)%*%phik
        fk<-ff[,k]
        if(length(Dargs$error.model)==1) {
          if(!is.na(match(Dargs$error.model,c("constant","exponential"))))
            resk<-sum((Dargs$yobs-fk)**2) else {
              if(Dargs$error.model=="proportional") {
                idx.okpred<-which(fk>.Machine$double.eps)
                vec<-(Dargs$yobs-fk)**2/cutoff(fk**2,.Machine$double.eps)
                resk<-sum(vec[idx.okpred])
                resk1<-sum(vec)
                } else resk<-0
            }
        } else resk<-0
        statr<-statr+resk
    }

print(resk)
```

```
## [1] 413.9428
```

```r
print(resk1)
```

```
## [1] 2.299501e+15
```

```r
kiter<-saemix.options$nbiter.burn+1
    # E-step
  xmcmc<-estep(kiter, Uargs, Dargs, opt, structural.model, mean.phi, varList, DYF, phiM)
  varList<-xmcmc$varList
  DYF<-xmcmc$DYF
  phiM<-xmcmc$phiM
  #  psiM<-transphi(phiM,saemix.model["transform.par"])

  #

  # M-step
```

```
#  if(opt$stepsize[kiter]>0) {
############# Stochastic Approximation
    xstoch<-mstep(kiter, Uargs, Dargs, opt, structural.model, DYF, phiM, varList, phi, betas, suffStat)
    varList<-xstoch$varList
    mean.phi<-xstoch$mean.phi
    phi<-xstoch$phi
    betas<-xstoch$betas
    suffStat<-xstoch$suffStat

    beta.I<-betas[Uargs$indx.betaI]
    fixed.psi<-transphi(matrix(beta.I,nrow=1),saemix.model["transform.par"])
    betaC<-betas[Uargs$indx.betaC]
    var.eta<-mydiag(varList$omega)
    l1<-betas.ini
    l1[Uargs$indx.betaI]<-fixed.psi
    l1[Uargs$indx.betaC]<-betaC

    if(Dargs$modeltype=="structural") {
      allpar[(kiter+1),]<-c(l1,var.eta[Uargs$i1.omega2],varList$pres[Uargs$ind.res])
    } else{
      allpar[(kiter+1),]<-c(l1,var.eta[Uargs$i1.omega2])
    }

  # } else { #end of loop on if (stepsize[kiter]>0)
  #   allpar[(kiter+1),]<-allpar[kiter,]
  # }
   if(Dargs$modeltype=="structural") {
      theta<-c(fixed.psi,var.eta[Uargs$i1.omega2],varList$pres[Uargs$ind.res])
    } else{
      theta<-c(fixed.psi,var.eta[Uargs$i1.omega2])
    }
  parpop[(kiter+1),]<-theta

print(theta)
print(head(phiM))
print(varList$pres)
```

```
saemixObject<-fit2b
```

```
kiter<-saemix.options$nbiter.burn+1

for (kiter in (saemix.options$nbiter.burn+1):(saemix.options$nbiter.sa)) { #
# Burn-in - resetting sufficient statistics
  if(opt$flag.fmin && kiter==saemix.options$nbiter.sa) {
    cat("Inside first loop, kiter=",kiter,":\n")
    Uargs$COV1<-Uargs$COV[,Uargs$ind.fix11]
    ind.prov<-!(varList$ind.eta %in% Uargs$i0.omega2)
    varList$domega2<-varList$domega2[ind.prov,ind.prov,drop=FALSE] # keep in domega2 only indices of pa
    varList$ind0.eta<-Uargs$i0.omega2
    varList$ind.eta<-1:(Uargs$nb.parameters)
    if(length(varList$ind0.eta)>0) varList$ind.eta<-varList$ind.eta[!(varList$ind.eta %in% varList$ind0
    Uargs$nb.etas<-length(varList$ind.eta)
    suffStat$statphi1<-0
    suffStat$statphi2<-0
```

```
    suffStat$statphi3<-0
  }

    # E-step
  xmcmc<-estep(kiter, Uargs, Dargs, opt, structural.model, mean.phi, varList, DYF, phiM)
  varList<-xmcmc$varList
  DYF<-xmcmc$DYF
  phiM<-xmcmc$phiM
  #  psiM<-transphi(phiM,saemix.model["transform.par"])

  # M-step
  if(opt$stepsize[kiter]>0) {
############# Stochastic Approximation
    xstoch<-mstep(kiter, Uargs, Dargs, opt, structural.model, DYF, phiM, varList, phi, betas, suffStat)
    varList<-xstoch$varList
    mean.phi<-xstoch$mean.phi
    phi<-xstoch$phi
    betas<-xstoch$betas
    suffStat<-xstoch$suffStat

    beta.I<-betas[Uargs$indx.betaI]
    fixed.psi<-transphi(matrix(beta.I,nrow=1),saemix.model["transform.par"])
    betaC<-betas[Uargs$indx.betaC]
    var.eta<-mydiag(varList$omega)
    l1<-betas.ini
    l1[Uargs$indx.betaI]<-fixed.psi
    l1[Uargs$indx.betaC]<-betaC

    if(Dargs$modeltype=="structural") {
      allpar[(kiter+1),]<-c(l1,var.eta[Uargs$i1.omega2],varList$pres[Uargs$ind.res])
    } else{
      allpar[(kiter+1),]<-c(l1,var.eta[Uargs$i1.omega2])
    }

  } else { #end of loop on if (stepsize[kiter]>0)
    allpar[(kiter+1),]<-allpar[kiter,]
  }
   if(Dargs$modeltype=="structural") {
      theta<-c(fixed.psi,var.eta[Uargs$i1.omega2],varList$pres[Uargs$ind.res])
    } else{
      theta<-c(fixed.psi,var.eta[Uargs$i1.omega2])
    }
  parpop[(kiter+1),]<-theta

# End of loop on kiter
}
print(theta)
print(head(phiM))
print(varList$pres)
```