Test individual functions in saemix 3.2

Emmanuelle

05/10/2020

Setup

- set up work directories
- two versions toggled by testMode
  - if testMode is FALSE, load the functions in R
  - if testMode is TRUE, use testthat functions

notes from Belhal (notes.txt in testbelhal)

- **1) for ORD data model, the response is a predictor. Test with new data without individual observations is non applicable.**
  - more of a comment than a question ?
- **2) For ORD data: problem in estimating parameters with new data (map and pop params) NEED TO DEBUG. Could be in map.saemix???**
  - **Eco** check on an example
  - Lucie's simulations seem to validate the estimation of the population parameters
  - Sofia's and my simulations show that individual parameters are poorly estimated in most models (but also true for Monolix)
- **3) COUNT data model: WHEN ONLY ONE PARAM TO ESTIMATE (fixed.estim=c(1,0)) OBTAIN:**
  - *Error in cbind(blocA, t(blocC)) :*
  - *le nombre de lignes des matrices doit correspondre (voir argument 2)*
  - => solved by Belhal on October 25

Classes

Data: SaemixData object

- testthat functions
  - **TODO** fix problems with testthat (probably call with helper functions)
  - normally should run through automatedTests_eco.R but doesn't work => look at how to set up automated testthat files
- testthat for classes
  - testeco/testthat_saemixData-class.R: works interactively (running tests one by one)
  - testbelhal/testthat_saemixData-class.R: works interactively
- testthat for read function (?)
  - testeco/testthat_saemixData-read.R: works interactively (running tests one by one)
  - testbelhal/testthat_saemixData-read.R: works interactively
    - removed the parts concerning continuous models

- summary function in testthat_summary.R works
- code below is the interactive version of testthat for data classes
- **TODO** test and check
  - update testeco/testthat_saemixData-covariates.R
  - update testeco/testthat_replaceData-cont.R (tested, works, but needs for theo.fit3 or theo.fit2 to be created before)
- **TODO**
  - silence the warnings "NA introduits"
  - problem reading binary data "Column name(s) do(es) not exist in the dataset, please check" (doesn't appear for TTE data)

```
# SaemixData class
## From data on disk
namtest<-"Creating SaemixData object from file on disk\n"
cat(namtest)
```

```
## Creating SaemixData object from file on disk
```

```
x<-try(saemixData(name.data=file.path(datDir,"theo.saemix.tab"),header=TRUE,sep=" ",na=NA, name.group=c
```

```
## Reading data from file /home/eco/work/saemix/saemixextension/data/theo.saemix.tab
## These are the first lines of the dataset as read into R. Please check the format of the data is appro
##   Id    Dose Time Concentration Weight Sex
## 1  1 319.992 0.25          2.84   79.6   1
## 2  1 319.992 0.57          6.57   79.6   1
## 3  1 319.992 1.12         10.50   79.6   1
## 4  1 319.992 2.02          9.66   79.6   1
## 5  1 319.992 3.82          8.58   79.6   1
## 6  1 319.992 5.10          8.36   79.6   1
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset /home/eco/work/saemix/saemixextension/data/theo.saemix.tab
##     Structured data: Concentration ~ Dose + Time | Id
##     X variable for graphs: Time (hr)
##     covariates: Weight (kg), Sex (-)
##       reference class for covariate Sex :  0
```

```
if(is(x, "try-error")) cat("Problem in",namtest) else theo.data<-x

## From data as a dataframe in the environment
namtest<-"Creating SaemixData object from dataframe\n"
cat(namtest)
```

```
## Creating SaemixData object from dataframe
```

```
theo.saemix<-read.table(file.path(datDir,"theo.saemix.tab"),header=T,na=".")
x<-try(saemixData(name.data=theo.saemix,header=TRUE,sep=" ",na=NA, name.group=c("Id"),name.predictors=c
```

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

```
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset theo.saemix
##     Structured data: Concentration ~ Dose + Time | Id
##     X variable for graphs: Time (hr)
##     covariates: Weight (kg), Sex (-)
##       reference class for covariate Sex :  0
```

```
if(is(x, "try-error")) cat("Problem in",namtest)

# SaemixRepData class
namtest<-"Creating SaemixRepData object\n"
cat(namtest)
```

```
## Creating SaemixRepData object
```

```
xrep<-new(Class="SaemixRepData",data=x)
print(xrep)
```

```
## Object of class saemixRepData
##     replicated data used in the SAEM algorithm
##     number of subjects in initial dataset 12
##     number of replications 1
##     number of subjects in replicated dataset 12
```

```
if(is(x, "try-error")) cat("Problem in",namtest)

# SaemixSimData class
namtest<-"Creating SaemixSimData object\n"
cat(namtest)
```

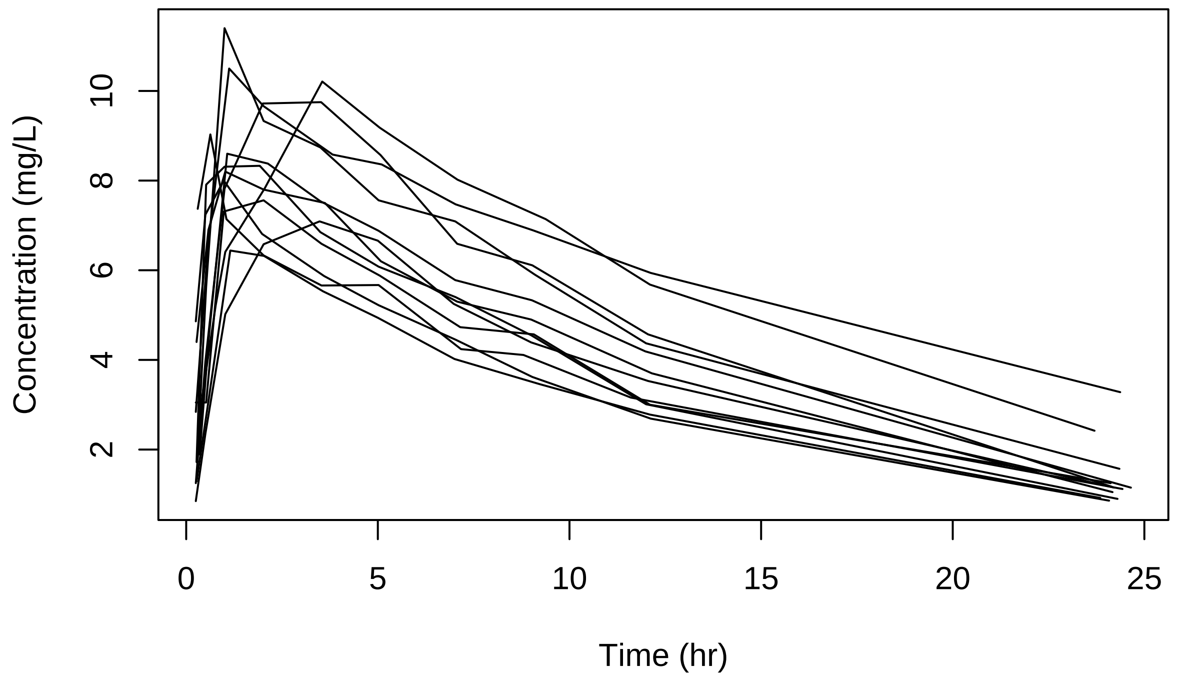
```
## Creating SaemixSimData object
```

```
xrep<-new(Class="SaemixSimData",data=x)
print(xrep)
```

```
## Object of class SaemixSimData
##     data simulated according to a non-linear mixed effect model
## Characteristics of original data
##     number of subjects: 12
##     summary of response:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.850   3.513   5.665   5.447   7.325  11.400
## Characteristics of simulated data
##     no simulations performed yet
```

```
if(is(x, "try-error")) cat("Problem in",namtest)
```

```

plot(theo.data)
```

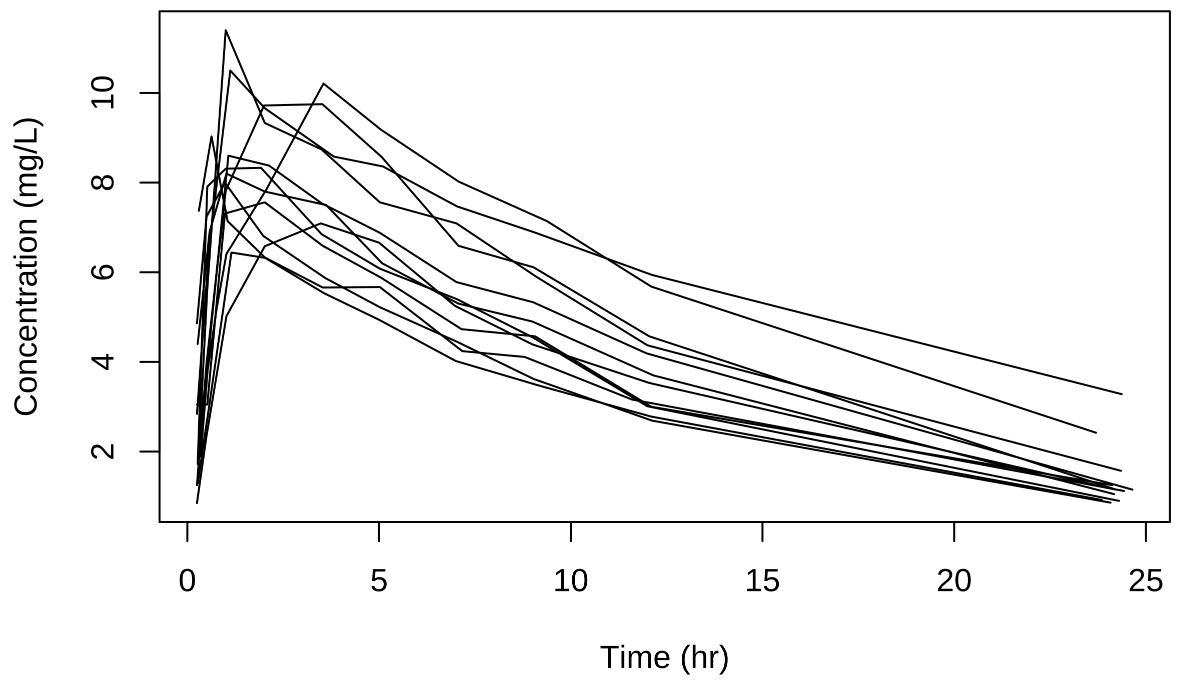


Plot data

```
theo.saemix2<-theo.saemix

theo.saemix2$Id <- as.character(theo.saemix2$Id)

theo.data2<-try(saemixData(name.data=theo.saemix2,header=TRUE,sep=" ",na=NA, name.group=c("Id"),name.pr
```

```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset theo.saemix2
##     Structured data: Concentration ~ Dose + Time | Id
##     X variable for graphs: Time (hr)
##     covariates: Weight (kg), Sex (-)
##       reference class for covariate Sex :  0
```

```
plot(theo.data2)
```



Model: SaemixModel object

Testing simple models

- Changes made
  - define modelType at the beginning of initialize to allow empty objects to be printed out
  - added a test for empty models in print to print out an appropriate message
- summary function in testthat_summary.R works
- split testthat file for SaemixModel to separate classes and methods
- check if function to validate covariance model works
  - changed name for consistency (no underscore)
  - added 3 quick tests in the testthat for SaemixModel
- predict function
  - predict from SaemixModel for psi taken from model object and for psi given as a vector
  - predict for different values of psi and predictors given to the function
  - added tests to testthat_saemixModel-function.R
- plot function to get predictions for a dataset based on a model and parameters (either from the model or a different set)
  - only works for continuous responses
  - uses ggplot so will need to add this as a dependency
- Belhal fixed the problem of the model needing at least 2 parameters to work with
- **TODO**
  - print function for empty models returns NULL, would rather it returned nothing
  - testthat function for the original plot function + documentation

```
# Empty model
namtest<-"Creating empty SaemixModel object\n"
cat(namtest)
```

```
## Creating empty SaemixModel object
```

```
xmod<-new(Class="SaemixModel")
print(xmod)
```

```
## Nonlinear mixed-effects model
## No model function set yet
```

```
## NULL
```

```
if(is(xmod, "try-error")) cat("Problem in",namtest)

# Minimal model
namtest<-"Creating minimal SaemixModel object\n"
cat(namtest)
```

```
## Creating minimal SaemixModel object
```

```
model1cpt<-function(psi,id,xidep) {
    dose<-xidep[,1]
    tim<-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)
  }
xmod<-saemixModel(model=model1cpt, psi0=matrix(c(1.,20,0.5), ncol=3,byrow=TRUE, dimnames=list(NULL, c("
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function
##   Model type:  structural
## function(psi,id,xidep) {
##     dose<-xidep[,1]
##     tim<-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)
##   }
##   Nb of parameters: 3
##       parameter names:  ka V CL
##       distribution:
##      Parameter Distribution Estimated
## [1,] ka        normal       Estimated
## [2,] V         normal       Estimated
## [3,] CL        normal       Estimated
##   Variance-covariance matrix:
##    ka V CL
## ka  1 0  0
## V   0 1  0
## CL   0 0  1
##   Error model: constant , initial values: a.1=1
##     No covariate in the model.
##     Initial values
##              ka  V  CL
## Pop.CondInit  1 20 0.5
```

```
if(is(xmod, "try-error")) cat("Problem in",namtest)

# Model with all elements
namtest<-"Creating full SaemixModel object\n"
cat(namtest)
```

```
## Creating full SaemixModel object
```

```
xmod<-saemixModel(model=model1cpt,description="One-compartment model with first-order absorption", psi0=
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption
##   Model type:  structural
## function(psi,id,xidep) {
##     dose<-xidep[,1]
##     tim<-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)
##   }
##   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  1
## CL  0 1  1
##   Error model: combined , initial values: a.1=1 b.1=0.5
##   Covariate model:
##      ka V CL
## [1,]  0 0  1
## [2,]  0 0  0
##     Initial values
##               ka  V    CL
## Pop.CondInit 1.0 20  0.50
## Cov.CondInit 0.1  0 -0.01
```

```r
if(is(xmod, "try-error")) cat("Problem in",namtest)
```

```r
# Binary model
binary.model<-function(psi,id,xidep) {
  tim<-xidep[,1]
  y<-xidep[,2]
  inter<-psi[id,1]
  slope<-psi[id,2]
  logit<-inter+slope*tim
  pevent<-exp(logit)/(1+exp(logit))
  pobs = (y==0)*(1-pevent)+(y==1)*pevent
  logpdf <- log(pobs)
  return(logpdf)
}

simulBinary<-function(psi,id,xidep) {
  tim<-xidep[,1]
  y<-xidep[,2]
  inter<-psi[id,1]
  slope<-psi[id,2]
  logit<-inter+slope*tim
  pevent<-1/(1+exp(-logit))
  ysim<-rbinom(length(tim),size=1, prob=pevent)
  return(ysim)
}
xmod<-saemixModel(model=binary.model,description="Binary model", modeltype="likelihood",
    psi0=matrix(c(-5,-.1,0,0),ncol=2,byrow=TRUE,dimnames=list(NULL,c("theta1","theta2"))),
    transform.par=c(0,0), covariate.model=c(0,1), covariance.model=matrix(c(1,0,0,0),ncol=2))
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  Binary model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   tim<-xidep[,1]
##   y<-xidep[,2]
##   inter<-psi[id,1]
##   slope<-psi[id,2]
##   logit<-inter+slope*tim
##   pevent<-exp(logit)/(1+exp(logit))
##   pobs = (y==0)*(1-pevent)+(y==1)*pevent
##   logpdf <- log(pobs)
##   return(logpdf)
## }
##   Nb of parameters: 2
##       parameter names:  theta1 theta2
##       distribution:
##      Parameter Distribution Estimated
## [1,] theta1    normal       Estimated
## [2,] theta2    normal       Estimated
##   Variance-covariance matrix:
```

```
##        theta1 theta2
## theta1      1      0
## theta2      0      0
##   Covariate model:
##      theta1 theta2
## [1,]      0      1
##     Initial values
##              theta1 theta2
## Pop.CondInit     -5   -0.1
## Cov.CondInit      0    0.0
```

```
xmod<-saemixModel(model=binary.model,description="Binary model", modeltype="likelihood", simulate.functi
    psi0=matrix(c(-5,-.1,0,0),ncol=2,byrow=TRUE,dimnames=list(NULL,c("theta1","theta2"))),
    transform.par=c(0,0), covariate.model=c(0,1), covariance.model=matrix(c(1,0,0,0),ncol=2))
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  Binary model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   tim<-xidep[,1]
##   y<-xidep[,2]
##   inter<-psi[id,1]
##   slope<-psi[id,2]
##   logit<-inter+slope*tim
##   pevent<-exp(logit)/(1+exp(logit))
##   pobs = (y==0)*(1-pevent)+(y==1)*pevent
##   logpdf <- log(pobs)
##   return(logpdf)
## }
##   Nb of parameters: 2
##       parameter names:  theta1 theta2
##       distribution:
##      Parameter Distribution Estimated
## [1,] theta1    normal       Estimated
## [2,] theta2    normal       Estimated
##   Variance-covariance matrix:
##        theta1 theta2
## theta1      1      0
## theta2      0      0
##   Covariate model:
##      theta1 theta2
## [1,]      0      1
##     Initial values
##              theta1 theta2
## Pop.CondInit     -5   -0.1
## Cov.CondInit      0    0.0
```

```
if(is(xmod, "try-error")) cat("Problem in",namtest)
```

```

theo.saemix<-read.table(file.path(datDir,"theo.saemix.tab"),header=T,na=".")
```

```
smx.data<-try(saemixData(name.data=theo.saemix,header=TRUE,sep=" ",na=NA, name.group=c("Id"),name.predi
```

Plot method for model+object

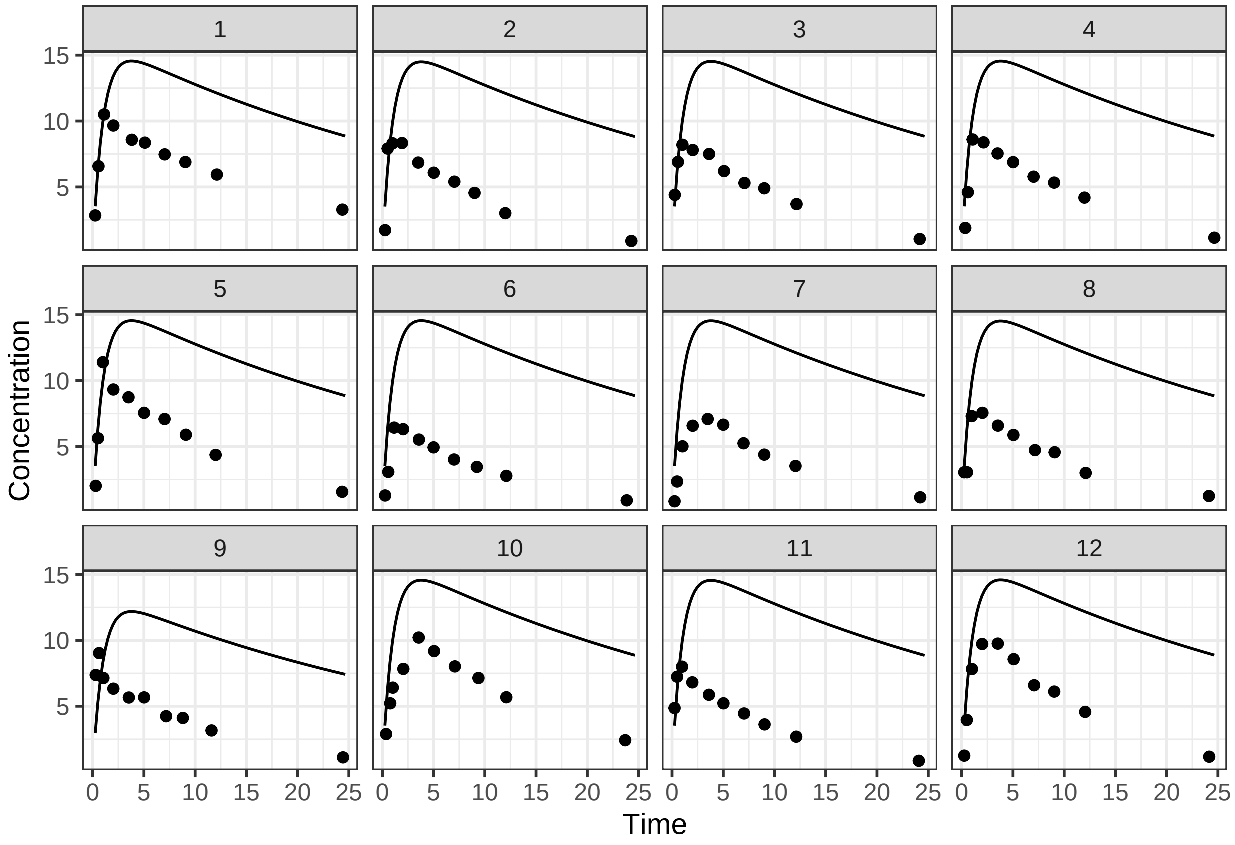
```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset theo.saemix
##     Structured data: Concentration ~ Dose + Time | Id
##     X variable for graphs: Time (hr)
##     covariates: Weight (kg), Sex (-)
##       reference class for covariate Sex :  0
```

```
xpred<-theo.saemix[,c("Dose","Time")]

model1cpt<-function(psi,id,xidep) {
    dose<-xidep[,1]
    tim<-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)
  }
xmod<-saemixModel(model=model1cpt, psi0=matrix(c(1.,20,0.5), ncol=3,byrow=TRUE, dimnames=list(NULL, c("
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function
##   Model type:  structural
## function(psi,id,xidep) {
##     dose<-xidep[,1]
##     tim<-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)
##   }
##   Nb of parameters: 3
##       parameter names:  ka V CL
##       distribution:
##      Parameter Distribution Estimated
## [1,] ka        normal       Estimated
## [2,] V         normal       Estimated
## [3,] CL        normal       Estimated
##   Variance-covariance matrix:
```

```
##    ka V CL
## ka  1 0  0
## V   0 1  0
## CL  0 0  1
##   Error model: constant , initial values: a.1=1
##     No covariate in the model.
##     Initial values
##                ka  V  CL
## Pop.CondInit  1 20 0.5
```

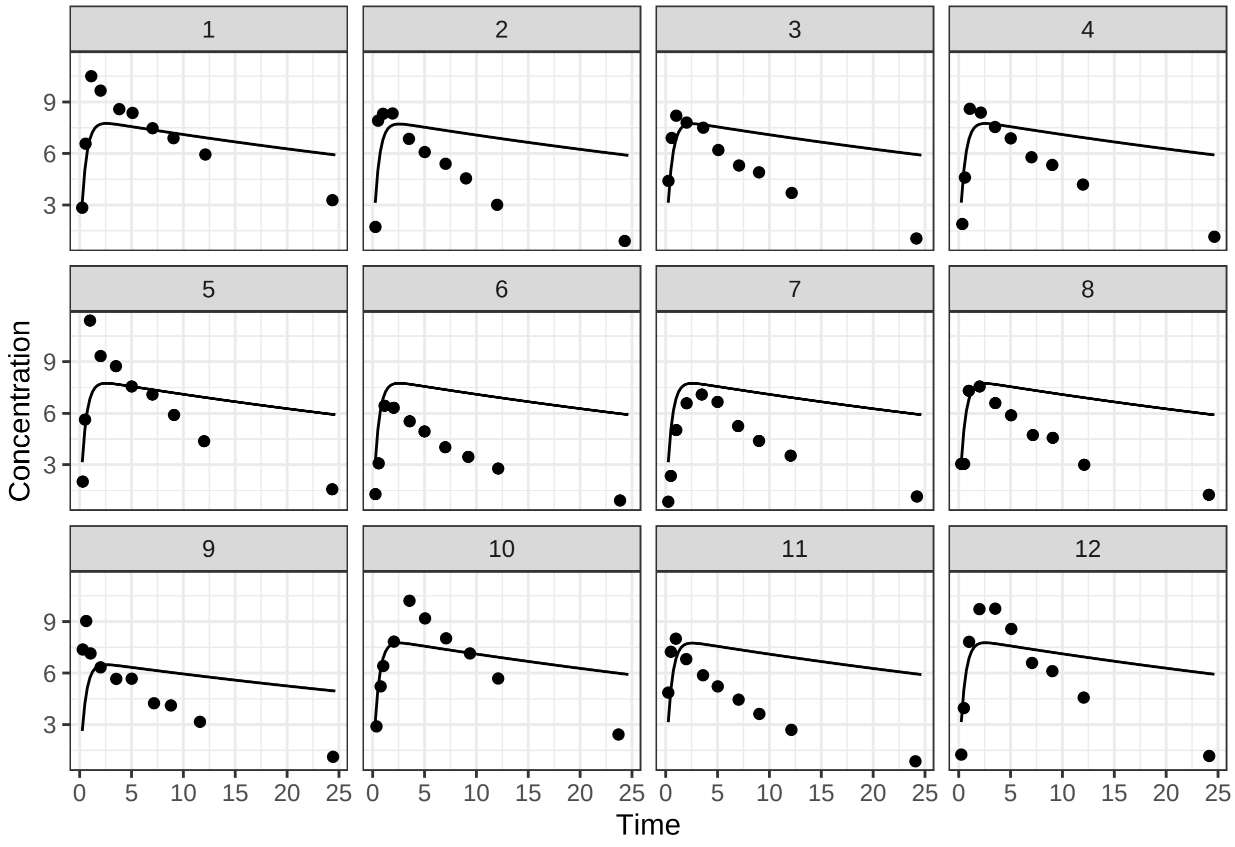
```
# With psi taken from the model
plot(xmod, smx.data)
```



```
head(predict(xmod, xpred)$predictions)
```

```
##   id    Dose Time      pred
## 1  1 319.992 0.25  3.527603
## 2  1 319.992 0.57  6.897479
## 3  1 319.992 1.12 10.602542
## 4  1 319.992 2.02 13.424870
## 5  1 319.992 3.82 14.555379
## 6  1 319.992 5.10 14.345433
```

```
# Psi given by user
plot(xmod, smx.data, psi=c(2, 40, 0.5))
```



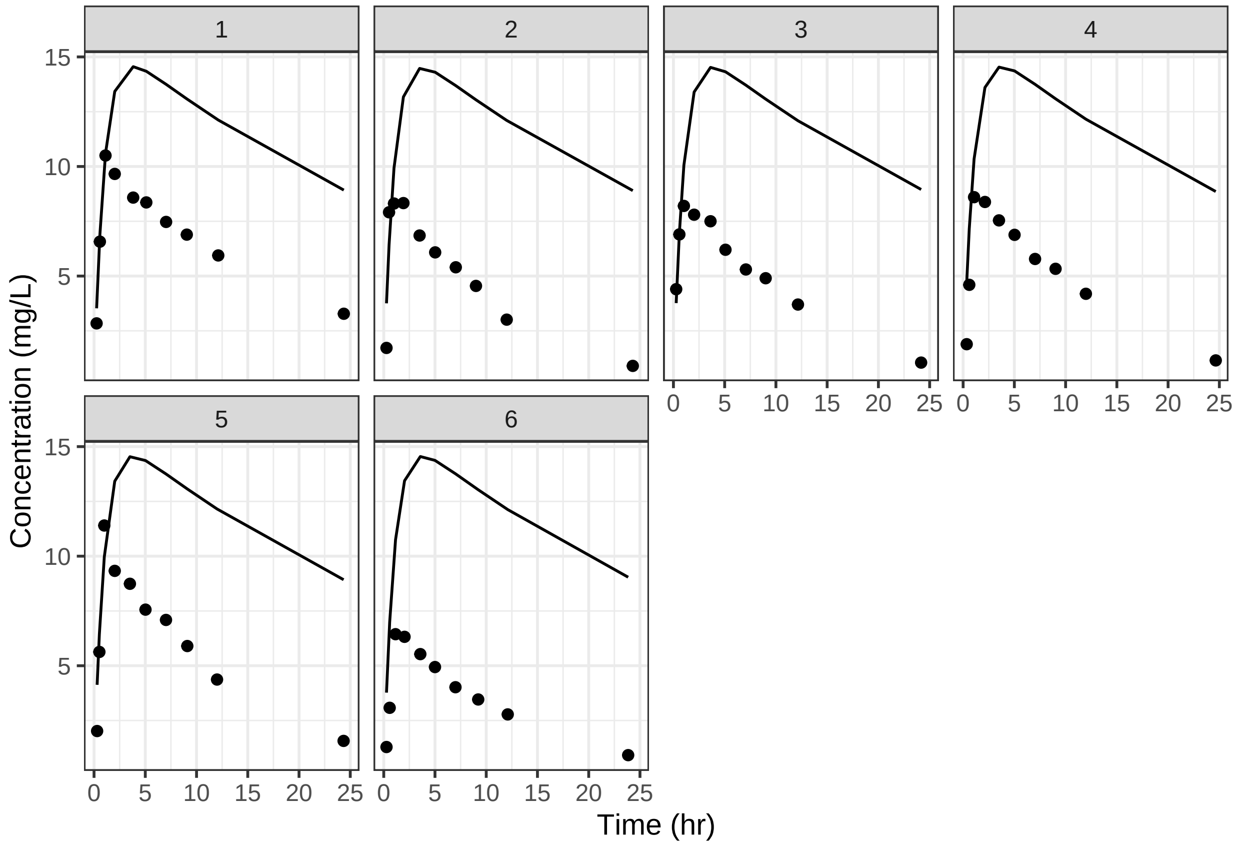
```
head(predict(xmod, xpred, psi=c(2, 40, 0.5))$predictions)
```

```
##   id    Dose Time     pred
## 1  1 319.992 0.25 3.142355
## 2  1 319.992 0.57 5.418381
## 3  1 319.992 1.12 7.081194
## 4  1 319.992 2.02 7.707731
## 5  1 319.992 3.82 7.670883
## 6  1 319.992 5.10 7.552635
```

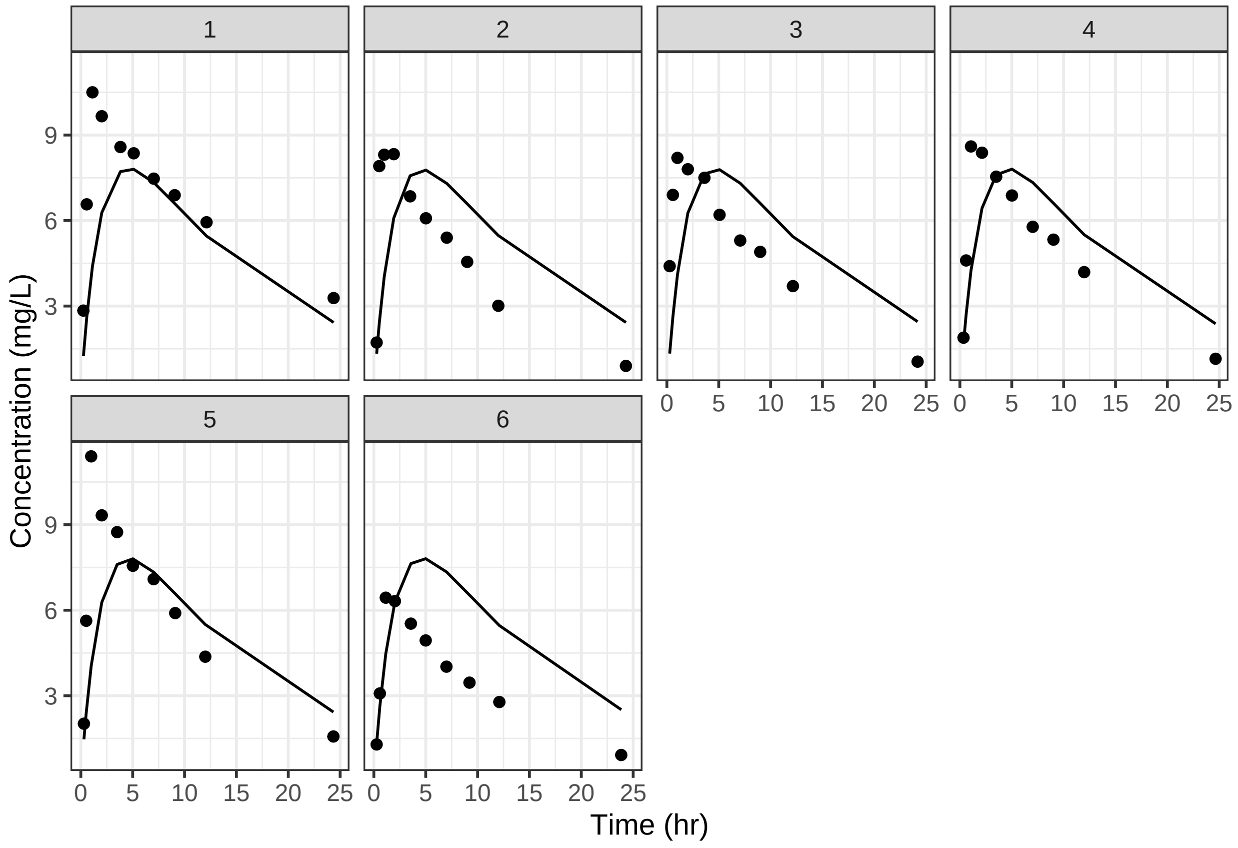
```
# Individual psi (same size as data, here N=12)
indpsi<-data.frame(ka=2, V=seq(25,47,2), CL=seq(2.5,4.7, 0.2))
head(predict(xmod, xpred, psi=indpsi)$predictions)
```

```
##   id    Dose Time      pred
## 1  1 319.992 0.25  4.968691
## 2  1 319.992 0.57  8.417811
## 3  1 319.992 1.12 10.611416
## 4  1 319.992 2.02 10.771907
## 5  1 319.992 3.82  9.189015
## 6  1 319.992 5.10  8.090185
```

```
# Check individual fixed effects
checkInitialFixedEffects(xmod, smx.data, id=c(1:6))
```



```r
checkInitialFixedEffects(xmod, smx.data, id=c(1:6), psi=c(0.5, 30, 2))
```



Results: SaemixRes object

- created testthat (short)
- added a test to vcov to handle empty objects
  - print, fitted, etc work as expected
  - added some messages for empty objects or not available types
- **TODO**
  - resid() or fitted() don't work, I need to use resid.SaemixRes, but I should be able to dispatch based on argument type like vcov
- **BUGFIX**
  - clarified the definitions of ypred and ppred (some mix-up had occurred): ppred=f(E(psi)) (predictions using the population parameters) and ypred=E(f(psi)) (expectancy of the individual predictions)

```
xres<-new(Class="SaemixRes")
print(xres)
```

```
## No fit performed yet.
```

```
## NULL
```

```
resid.SaemixRes(xres)
```

```
## No residuals of type ires available
```

```
fitted.SaemixRes(xres)
```

```
## No fitted values of type ipred available
```

```
vcov(xres)
```

```
## NULL
```

Fitted object: SaemixObject object

- class
- summary function
  - tests in **testthat_summary.R** currently failed (list size has increased, probably becuase of the changes to the statistical criterion) **TODO**
- simulated annealing
  - default control options changed to have SA only on K1/2 iterations (bug previously where SA would be for all K1 iterations, with poorer estimates as a result)
  - checked that SA is also applied to residual error models (modification suggested by Edouard Ollier, november 2016)

```
# Control options
xopt<-saemixControl()
cat("K1=",xopt$nbiter.saemix," nb.SA=",xopt$nbiter.sa,"\n")
```

```
## K1= 300 100  nb.SA= 150
```

```
# Empty object
smx.data<-saemixData(name.data=file.path(datDir,"theo.saemix.tab"),header=T,na=".", name.group=c("Id"),r
model1cpt<-function(psi,id,xidep) {
  dose<-xidep[,1]
  tim<-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)
}
smx.model<-saemixModel(model=model1cpt,description="One-compartment model with first-order absorption",
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption
##   Model type:  structural
## function(psi,id,xidep) {
##   dose<-xidep[,1]
##   tim<-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)
## }
##   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  1
## CL  0 1  1
##   Error model: combined , initial values: a.1=1 b.1=0.5
##   Covariate model:
##      ka V CL
## [1,]  0 0  1
## [2,]  0 0  0
##     Initial values
##               ka  V    CL
## Pop.CondInit 1.0 20  0.50
## Cov.CondInit 0.1  0 -0.01
```

```
smx.opt<-saemixControl(nb.chains=5,nbiter.saemix = c(500,300), ipar.lmcmc = 100)
x<-createSaemixObject.empty(smx.model,smx.data,smx.opt)
print(x)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data             ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset /home/eco/work/saemix/saemixextension/data/theo.saemix.tab
##     Structured data: Concentration ~ Dose + Time | Id
##     X variable for graphs: Time (hr)
##     covariates: Weight (-), Sex (-)
##       reference class for covariate Sex :  0
## Dataset characteristics:
##     number of subjects:     12
##     number of observations: 120
##     average/min/max nb obs: 10.00  /  10  /  10
## First 10 lines of data:
##    Id    Dose  Time Concentration Weight Sex mdv cens occ ytype
## 1   1 319.992  0.25          2.84   79.6   1   0    0   1     1
## 2   1 319.992  0.57          6.57   79.6   1   0    0   1     1
## 3   1 319.992  1.12         10.50   79.6   1   0    0   1     1
## 4   1 319.992  2.02          9.66   79.6   1   0    0   1     1
## 5   1 319.992  3.82          8.58   79.6   1   0    0   1     1
## 6   1 319.992  5.10          8.36   79.6   1   0    0   1     1
## 7   1 319.992  7.03          7.47   79.6   1   0    0   1     1
## 8   1 319.992  9.05          6.89   79.6   1   0    0   1     1
## 9   1 319.992 12.12          5.94   79.6   1   0    0   1     1
## 10  1 319.992 24.37          3.28   79.6   1   0    0   1     1
## -----------------------------------
## ----          Model            ----
## -----------------------------------
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption
##   Model type:  structural
## function(psi,id,xidep) {
##   dose<-xidep[,1]
##   tim<-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)
## }
##   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  1
## CL  0 1  1
##   Error model: combined , initial values: a.1=1 b.1=0.5
##   Covariate model:
##        ka V CL
## Weight  0 0  1
## Sex     0 0  0
##     Initial values
##              ka  V    CL
## Pop.CondInit 1.0 20  0.50
## Cov.CondInit 0.1  0 -0.01
## ----------------------------------
## ----    Key algorithm options  ----
## ----------------------------------
##     Estimation of individual parameters (MAP)
##     Estimation of standard errors and linearised log-likelihood
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=500, K2=300
##     Number of chains:  5
##     Seed:  23456
##     Number of MCMC iterations for IS:  5000
##     Simulations:
##         nb of simulated datasets used for npde:  1000
##         nb of simulated datasets used for VPC:  100
##     Input/output
##         save the results to a file:  TRUE
##         save the graphs to files:  TRUE
##         directory where results should be saved:  newdir
## ----------------------------------------------------
## ----                  Results                   ----
## No fit performed yet.
## NULL
```

```r
show(x)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------------
## ----          Data and Model          ----
## -----------------------------------------
## Data
##     Dataset /home/eco/work/saemix/saemixextension/data/theo.saemix.tab
##     Longitudinal data: Concentration ~ Dose + Time | Id
##
## Model:
##     One-compartment model with first-order absorption
##      3 parameters: ka V CL
##      error model: combined
##      covariate model:
##        ka V CL
## Weight  0 0  1
## Sex     0 0  0
##
## Key options
##     Estimation of individual parameters (MAP)
##     Estimation of standard errors and linearised log-likelihood
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=500, K2=300
##     Number of chains:  5
##     Seed:  23456
##     Number of MCMC iterations for IS:  5000
##     Input/output
##         save the results to a file:  TRUE
##         save the graphs to files:  TRUE
##         directory where results are saved:  newdir
```

Auxiliary functions

error type, error, ssq

- moved to combined 2 error model for residual error ($g^2 = a^2 + b^2 f^2$)
  - **TODO** add to documentation
  - added to CHANGES
- modified testthat_ssq.R to create testthat_ssq_combined2.R

NLMEM fits

Continuous response model

Main fit

- Theophylline data

```r
# Theophylline data, base model
theo.saemix<-read.table(file.path(datDir,"theo.saemix.tab"),header=T)
saemix.data<-saemixData(name.data=theo.saemix,header=TRUE,sep=" ",na=NA,
                        name.group=c("Id"),name.predictors=c("Dose","Time"),
                        name.response=c("Concentration"),name.covariates=c("Weight","Sex"),
                         units=list(x="hr",y="mg/L",covariates=c("kg","-")), name.X="Time", verbose = FAL

model1cpt<-function(psi,id,xidep) {
  dose<-xidep[,1]
  tim<-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)
}

# Model with covariate Weight
saemix.model<-saemixModel(model=model1cpt,modeltype="structural",
                          description="One-compartment model with first-order absorption",
                          psi0=matrix(c(1.,20,0.5,0.1,0,-0.01),ncol=3,byrow=TRUE, dimnames=list(NULL, c
                          transform.par=c(1,1,1),covariate.model=matrix(c(0,0,1,0,0,0),ncol=3,byrow=TRUE

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE)
saemix.fit<-saemix(saemix.model,saemix.data,saemix.options)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data              ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset theo.saemix
##     Structured data: Concentration ~ Dose + Time | Id
##     X variable for graphs: Time (hr)
##     covariates: Weight (kg), Sex (-)
##       reference class for covariate Sex :  0
## Dataset characteristics:
##     number of subjects:     12
##     number of observations: 120
##     average/min/max nb obs: 10.00  /  10  /  10
## First 10 lines of data:
##    Id    Dose  Time Concentration Weight Sex mdv cens occ ytype
## 1   1 319.992  0.25          2.84   79.6   1   0    0   1     1
## 2   1 319.992  0.57          6.57   79.6   1   0    0   1     1
## 3   1 319.992  1.12         10.50   79.6   1   0    0   1     1
## 4   1 319.992  2.02          9.66   79.6   1   0    0   1     1
## 5   1 319.992  3.82          8.58   79.6   1   0    0   1     1
## 6   1 319.992  5.10          8.36   79.6   1   0    0   1     1
## 7   1 319.992  7.03          7.47   79.6   1   0    0   1     1
## 8   1 319.992  9.05          6.89   79.6   1   0    0   1     1
## 9   1 319.992 12.12          5.94   79.6   1   0    0   1     1
## 10  1 319.992 24.37          3.28   79.6   1   0    0   1     1
## -----------------------------------
## ----          Model             ----
## -----------------------------------
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption
##   Model type:  structural
## function(psi,id,xidep) {
##   dose<-xidep[,1]
##   tim<-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: 0x55eb221f5990>
##   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: constant , initial values: a.1=1
##   Covariate model:
##        [,1] [,2] [,3]
## Weight    0    0    1
##     Initial values
##              ka  V    CL
## Pop.CondInit 1.0 20  0.50
## Cov.CondInit 0.1  0 -0.01
## ----------------------------------
## ----    Key algorithm options  ----
## ----------------------------------
##     Estimation of individual parameters (MAP)
##     Estimation of standard errors and linearised log-likelihood
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=300, K2=100
##     Number of chains:  5
##     Seed:  632545
##     Number of MCMC iterations for IS:  5000
##     Simulations:
##         nb of simulated datasets used for npde:  1000
##         nb of simulated datasets used for VPC:  100
##     Input/output
##         save the results to a file:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter       Estimate SE    CV(%) p-value
## [1,] ka                1.573   0.300  19.1 -
## [2,] V                31.524   1.410   4.5 -
## [3,] CL                1.587   1.005  63.3 -
## [4,] beta_Weight(CL)  0.008   0.009 113.3 0.38
## [5,] a.1               0.742   0.057   7.7 -
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate SE     CV(%)
## ka omega2.ka 0.385    0.1738 45
## V  omega2.V  0.016    0.0094 58
## CL omega2.CL 0.068    0.0333 49
## ----------------------------------------------------
## ------  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= 343.4026
##       AIC = 359.4026
##       BIC = 363.2818
##
## Likelihood computed by importance sampling
##       -2LL= 344.6988
##       AIC = 360.6988
##       BIC = 364.5781
## ----------------------------------------------------
```

```r
# Model with 2 covariates and a covariance model
saemix.model3<-saemixModel(model=model1cpt,modeltype="structural",
          description="One-compartment model with first-order absorption",
          psi0=matrix(c(1.,20,0.5,0.1,0,-0.01),ncol=3,byrow=TRUE, dimnames=list(NULL, c("ka","V","CL"))
          covariance.model=matrix(c(1,0,0,0,1,1,0,1,1),ncol=3,byrow=TRUE),
          covariate.model=matrix(c(0,0,1,0,1,0),ncol=3,byrow=TRUE),
          transform.par=c(1,1,1),error.model="proportional")
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption
##   Model type:  structural
## function(psi,id,xidep) {
##   dose<-xidep[,1]
##   tim<-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: 0x55eb221f5990>
##   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  1
## CL  0 1  1
##   Error model: proportional , initial values: b.1=1
##   Covariate model:
##      ka V CL
## [1,]  0 0  1
## [2,]  0 1  0
##     Initial values
##               ka  V    CL
## Pop.CondInit 1.0 20  0.50
## Cov.CondInit 0.1  0 -0.01
```

```
saemix.options<-list(seed=12345,save=FALSE,save.graphs=FALSE, displayProgress=FALSE)
saemix.fit3<-saemix(saemix.model3,saemix.data,saemix.options)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data             ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset theo.saemix
##     Structured data: Concentration ~ Dose + Time | Id
##     X variable for graphs: Time (hr)
##     covariates: Weight (kg), Sex (-)
##       reference class for covariate Sex :  0
## Dataset characteristics:
##     number of subjects:     12
##     number of observations: 120
##     average/min/max nb obs: 10.00  /  10  /  10
## First 10 lines of data:
##    Id    Dose  Time Concentration Weight Sex mdv cens occ ytype
## 1   1 319.992  0.25          2.84   79.6   1   0    0   1     1
## 2   1 319.992  0.57          6.57   79.6   1   0    0   1     1
## 3   1 319.992  1.12         10.50   79.6   1   0    0   1     1
## 4   1 319.992  2.02          9.66   79.6   1   0    0   1     1
## 5   1 319.992  3.82          8.58   79.6   1   0    0   1     1
## 6   1 319.992  5.10          8.36   79.6   1   0    0   1     1
## 7   1 319.992  7.03          7.47   79.6   1   0    0   1     1
## 8   1 319.992  9.05          6.89   79.6   1   0    0   1     1
## 9   1 319.992 12.12          5.94   79.6   1   0    0   1     1
## 10  1 319.992 24.37          3.28   79.6   1   0    0   1     1
## -----------------------------------
## ----          Model            ----
## -----------------------------------
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption
##   Model type:  structural
## function(psi,id,xidep) {
##   dose<-xidep[,1]
##   tim<-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: 0x55eb221f5990>
##   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  1
## CL  0 1  1
##   Error model: proportional , initial values: b.1=1
##   Covariate model:
##        [,1] [,2] [,3]
## Weight    0    0    1
## Sex       0    1    0
##     Initial values
##               ka  V    CL
## Pop.CondInit 1.0 20  0.50
## Cov.CondInit 0.1  0 -0.01
## psi1         0.1  0 -0.01
## -----------------------------------
## ----    Key algorithm options  ----
## -----------------------------------
##     Estimation of individual parameters (MAP)
##     Estimation of standard errors and linearised log-likelihood
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=300, K2=100
##     Number of chains:  5
##     Seed:  12345
##     Number of MCMC iterations for IS:  5000
##     Simulations:
##         nb of simulated datasets used for npde:  1000
##         nb of simulated datasets used for VPC:  100
##     Input/output
##         save the results to a file:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter       Estimate SE     CV(%) p-value
## [1,] ka               1.4864  0.3016  20.3 -
## [2,] V               30.5095  1.9290   6.3 -
## [3,] beta_Sex(V)      0.0816  0.0725  88.9 0.26
## [4,] CL               3.9037  1.7707  45.4 -
## [5,] beta_Weight(CL) -0.0049  0.0064 130.7 0.44
## [6,] b.1              0.1597  0.0122   7.6 -
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##       Parameter Estimate SE    CV(%)
## ka    omega2.ka 0.441    0.193 44
## V     omega2.V  0.017    0.010 61
## CL    omega2.CL 0.063    0.028 44
## covar cov.V.CL  0.033    0.015 47
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.ka omega2.V omega2.CL
## omega2.ka 1         0        0
## omega2.V  0         1        1
## omega2.CL 0         1        1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= 333.5813
##       AIC = 353.5813
##       BIC = 358.4304
##
## Likelihood computed by importance sampling
##       -2LL= 349.1338
##       AIC = 369.1338
##       BIC = 373.9829
## ----------------------------------------------------
```

```r
theo.fit1<-saemix.fit
theo.fit3<-saemix.fit3

if(FALSE) { # done later
  # MAP
theo.fit1<-map.saemix(theo.fit1)

# Predictions - ipred, population predictions using MAP
theo.fit1<-saemix.predict(theo.fit1, type=c("ipred","ppred"))

# Conditional distributions
theo.fit1<-conddist.saemix(theo.fit1)

# Predictions - icpred, population predictions using conditional distributions
theo.fit1<-saemix.predict(theo.fit1, type=c("icpred"))

# Simulate
theo.fit1<-simulate(theo.fit1)

# Residuals
theo.fit1<-compute.sres(theo.fit1)

# Predictions - population predictions as E(f())
theo.fit1<-saemix.predict(theo.fit1, type=c("ypred"))

# Residuals
theo.fit1<-compute.sres(theo.fit1)

# Data and convergence plots
plot(theo.fit1, plot.type="convergence")
plot(theo.fit1, plot.type="likelihood")
plot(theo.fit1, plot.type="data")

# Random effects
plot(theo.fit1, plot.type="random.effects")
plot(theo.fit1, plot.type="correlations")
plot(theo.fit1, plot.type="marginal.distribution")


# Default plots
plot(theo.fit1)

# Diagnostics
plot(theo.fit1, plot.type="observations.vs.predictions")
plot(theo.fit1, plot.type="individual.fit")
plot(theo.fit1, plot.type="population.fit")
plot(theo.fit1, plot.type="both.fit")
plot(theo.fit1, plot.type="vpc")
}
```

Individual parameters and predictions

- predict function
  - **TODO** check why we don't have predictions at the end of the fit ? We should have them by default (from MAP at least for individual predictions and for ypred for population predictions)

```
saemixObject<-theo.fit1

# Conditional distribution
myfit <- conddist.saemix(saemixObject, nsamp = 100)
dim(myfit@results@psi.samp)
```

```
## [1]  12   3 100
```

```
# Predictions for the observations in the original data
## Extract predictions - empty for the moment (?)
```

```r
vec<-predict(saemixObject)
vec<-predict(saemixObject, type="ypred")
```

```
## No fitted values of type ypred available

## Simulating data using nsim = 1000 simulated datasets
## Computing WRES and npde ..
```

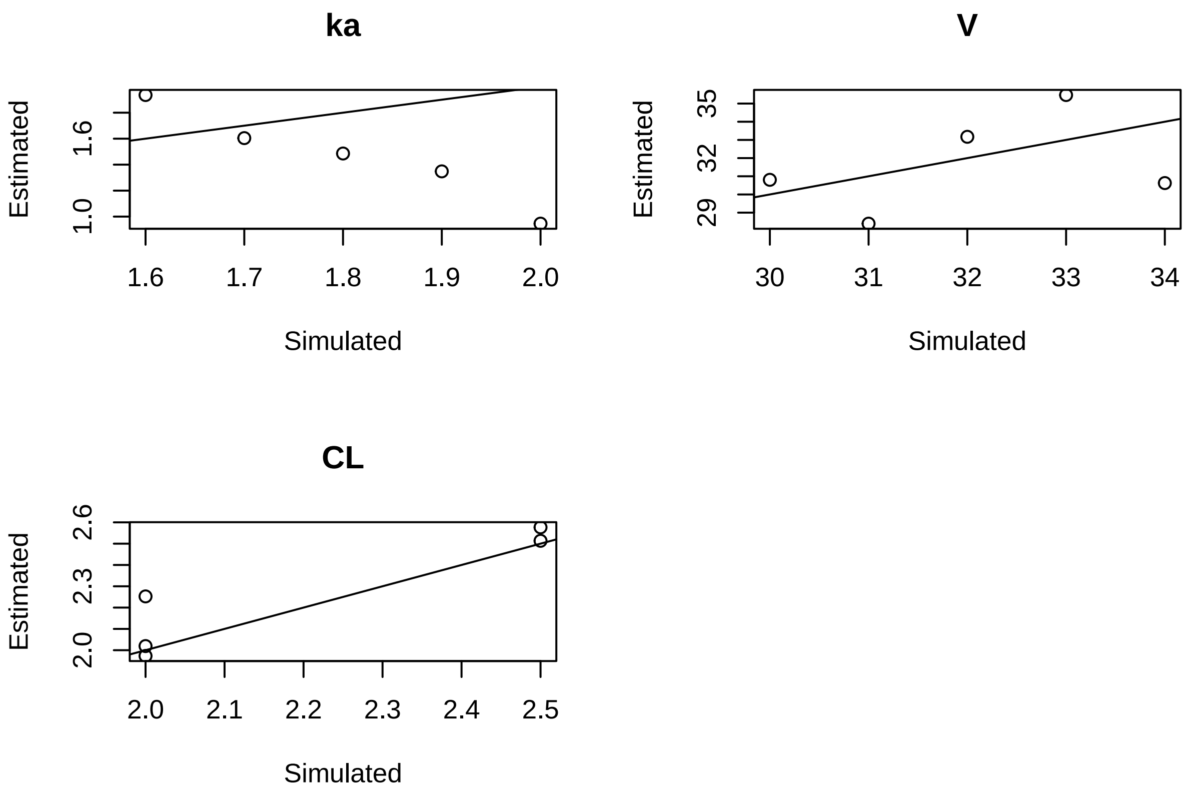
```r
# Fit then extract predictions
fit.pred<-saemix.predict(saemixObject)
```

```
## Simulating data using nsim = 1000 simulated datasets
## Computing WRES and npde ..
```

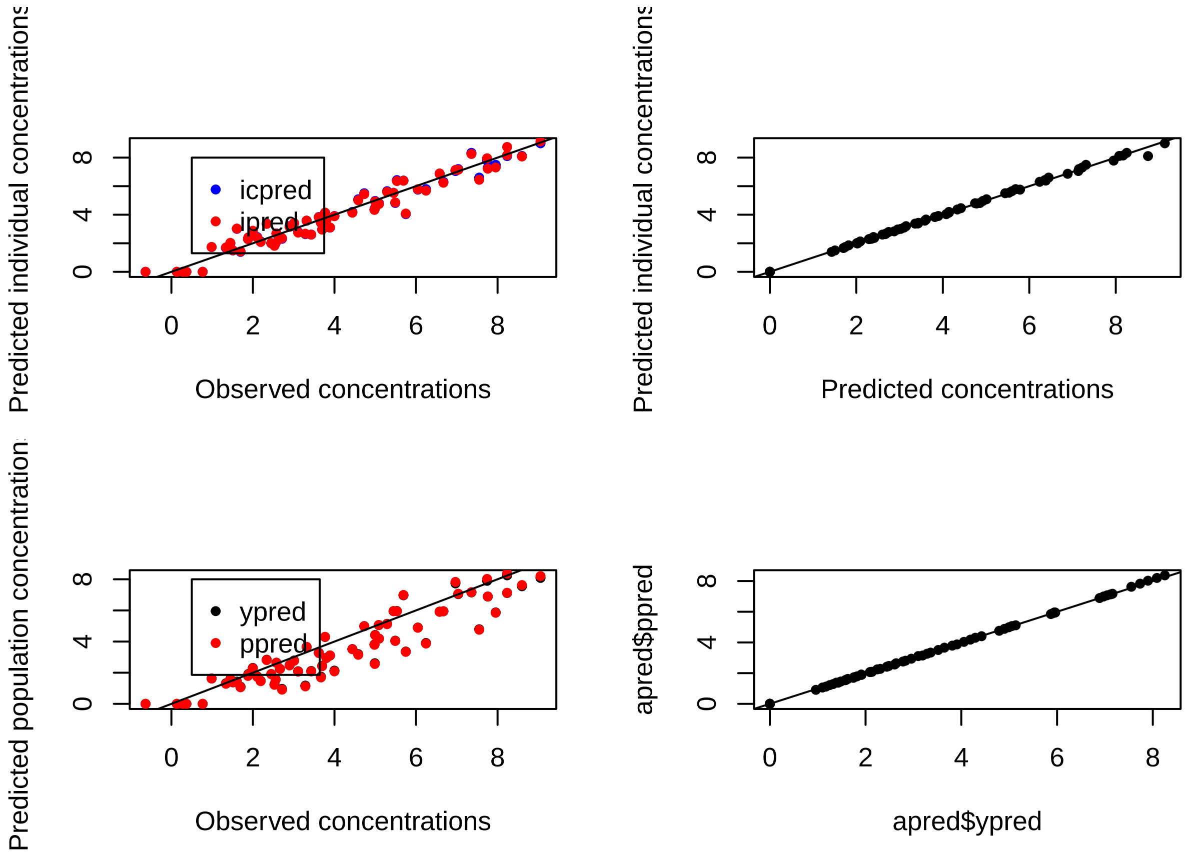
```r
predict(fit.pred)
```

```
##   [1] 3.8361716 6.7770612 9.0453887 9.7915773 9.0824875 8.4180687 7.4909441
##   [8] 6.6284085 5.5037789 2.6209359 3.9937931 6.1363488 8.0051945 8.4432717
##  [15] 7.4193818 6.3821842 5.2194385 4.2852271 3.1735165 0.9263284 4.3341624
##  [22] 6.8074403 8.1469815 8.2836718 7.3107810 6.4548006 5.4445827 4.6160310
##  [29] 3.5257537 1.2610461 3.4391588 5.0831921 6.9762892 8.2944330 7.9620006
##  [36] 7.0972344 5.9864215 5.0341750 3.8936438 1.2962915 4.2090935 6.1973169
##  [43] 8.5909108 9.7337694 9.0191273 7.9841382 6.7572127 5.6773281 4.4533590
##  [50] 1.5833842 2.0527692 3.7274756 5.5220302 6.4853334 6.2283441 5.4738437
##  [57] 4.4330134 3.4803094 2.5387570 0.7004470 1.5838733 2.8648349 4.7740623
##  [64] 6.5715834 7.0381999 6.5539482 5.6011228 4.6667618 3.5077998 1.1113723
##  [71] 2.5721438 4.4758982 6.3538580 7.5387984 7.0392285 6.1659766 5.0660080
##  [78] 4.2263351 3.1744892 1.0199796 6.9342108 7.9300490 7.8042648 7.1398069
##  [85] 6.1951953 5.3856307 4.4002610 3.7752484 2.9017224 0.8688890 2.9279027
##  [92] 5.1888895 6.2382307 8.6370674 9.3364372 8.9040112 7.9179543 6.8036525
##  [99] 5.6580852 2.5663961 4.8963427 6.9273209 7.9185371 7.4719685 6.3520819
## [106] 5.5006674 4.4868312 3.6635676 2.6784857 0.7969607 2.5840958 4.5539021
## [113] 7.1561695 9.2495965 9.2972432 8.3652775 7.0479362 5.9120712 4.4997449
## [120] 1.5053295
```

```r
# Predictions for the observations in a new dataset
## Create a new dataset
xtim<-seq(0,24,2)
nsuj<-5
xwei<-seq(50,90,length.out = nsuj)
xsex<-rep(c("F","M"),length.out=nsuj)
xdose<-seq(280,320,length.out=nsuj)
theo.newdata<-data.frame(Id=rep(1:nsuj,each=length(xtim)),Time=rep(xtim,nsuj),Dose=rep(xdose,each=lengt

psiM<-data.frame(ka=seq(1.6,2,0.1),V=seq(34,30),CL=c(2,2.5,2,2.5,2))
fpred<-saemixObject["model"]["model"](psiM, theo.newdata$Id, theo.newdata[,c("Dose","Time")])
theo.newdata$Concentration<-fpred+rnorm(length(fpred),sd=0.74)
theo.psiM<-psiM
test.newdata<-theo.newdata

## Use predict function
if(FALSE) { # debugging func_estimParam.R
  newdata<-theo.newdata
  type=c("ipred", "ypred", "ppred", "icpred")
  nsamp=10
  # Within functions, debug estimateIndividualParametersNewdata
  ctype <- c("mode","mean")
  yfit1<-estimateIndividualParametersNewdata(saemixObject,type=ctype,nsamp=nsamp)

  yfit1 <- map.saemix(saemixObject)

  yfit1 <- compute.eta.map(saemixObject)

  smx.repl<-replaceData.saemixObject(saemixObject,theo.newdata)
  smx.repl@data@N
  smx.repl<-estimateMeanParametersNewdata(smx.repl)

  newdata<-smx.repl["data"]
  chdat<-smx.repl["rep.data"]
  NM<-chdat["NM"]
  IdM<-chdat["dataM"]$IdM
  yM<-chdat["dataM"]$yM
  XM<-chdat["dataM"][,c(newdata["name.predictors"],newdata["name.cens"],newdata["name.mdv"],newdata["na
  mean.phi<-smx.repl["results"]["mean.phi"]
  psiM<-transphi(mean.phi,smx.repl["model"]["transform.par"])
  fpred<-smx.repl["model"]["model"](psiM, IdM, XM)
  colnames(psiM)<-smx.repl["model"]["name.modpar"]
  predictions<-data.frame(IdM,XM,ppred=fpred)
  colnames(predictions)[1]<-newdata["name.group"]
  parameters<-list(id=unique(newdata["data"][,newdata["name.group"]]), population=psiM)
  smx.repl["results"]["ppred"]<-fpred

  saemixObject <- smx.repl

  smx.repl<-estimateIndividualParametersNewdata(smx.repl,type=c("mode"),nsamp=nsamp)
  yfit1 <- compute.eta.map(smx.repl)

}

pred1<-predict(saemixObject, theo.newdata, type=c("ipred", "ypred", "ppred", "icpred"))

mylist<-saemixPredictNewdata(saemixObject, theo.newdata, type=c("ipred", "ypred", "ppred", "icpred"))
param<-mylist$param$map.psi
par(mfrow=c(2,2))
for(i in 1:3) {
  plot(theo.psiM[,i],param[,i],main=colnames(psiM)[i],xlab="Simulated",ylab="Estimated")
  abline(0,1)
}

apred<-mylist$predictions
par(mfrow=c(2,2))
```



```r
plot(theo.newdata$Concentration,apred$ipred,pch=20,col="Blue", xlab="Observed concentrations", ylab="Pr
points(theo.newdata$Concentration,apred$icpred,pch=20,col="Red", xlab="Observed concentrations", ylab="
abline(0,1)
legend(0.5,8,pch=20,col=c("Blue","Red"),c("icpred","ipred"))

plot(apred$icpred,apred$ipred,pch=20,col="Black", xlab="Predicted concentrations", ylab="Predicted indi
abline(0,1)

plot(theo.newdata$Concentration,apred$ypred,pch=20,col="Black", xlab="Observed concentrations", ylab="P
points(theo.newdata$Concentration,apred$ppred,pch=20,col="Red", xlab="Observed concentrations", ylab="P
abline(0,1)
legend(0.5,8,pch=20,col=c("Black","Red"),c("ypred","ppred"))

plot(apred$ypred,apred$ppred,pch=20,col="Black")
abline(0,1)
```

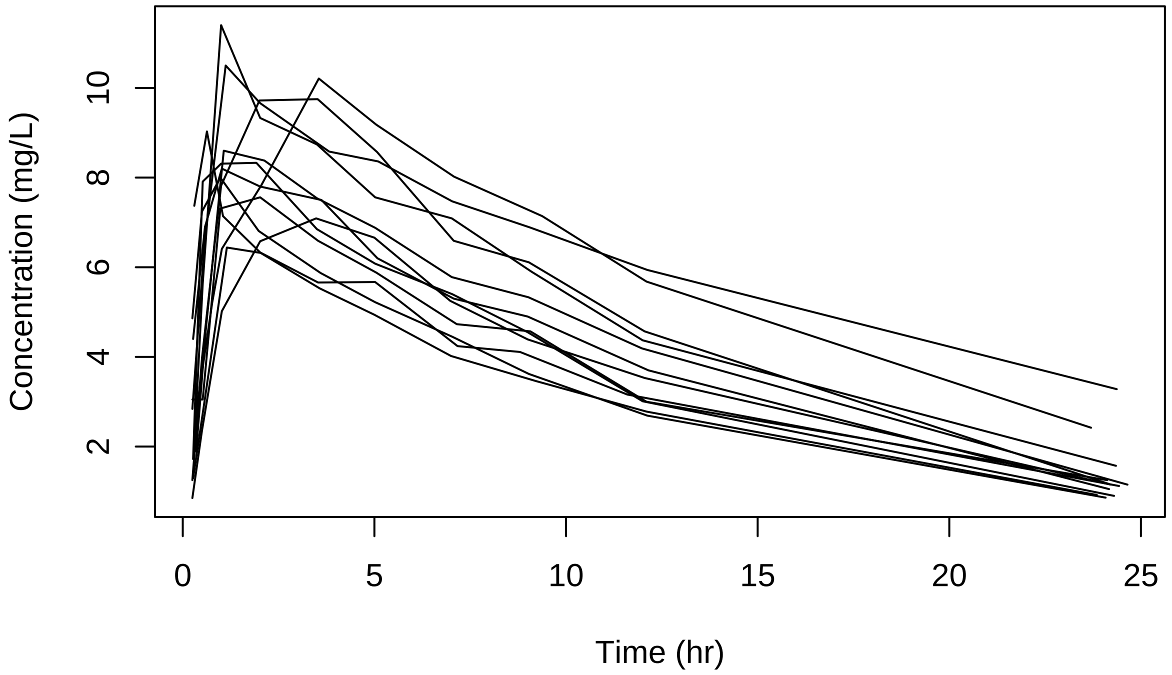


Plots

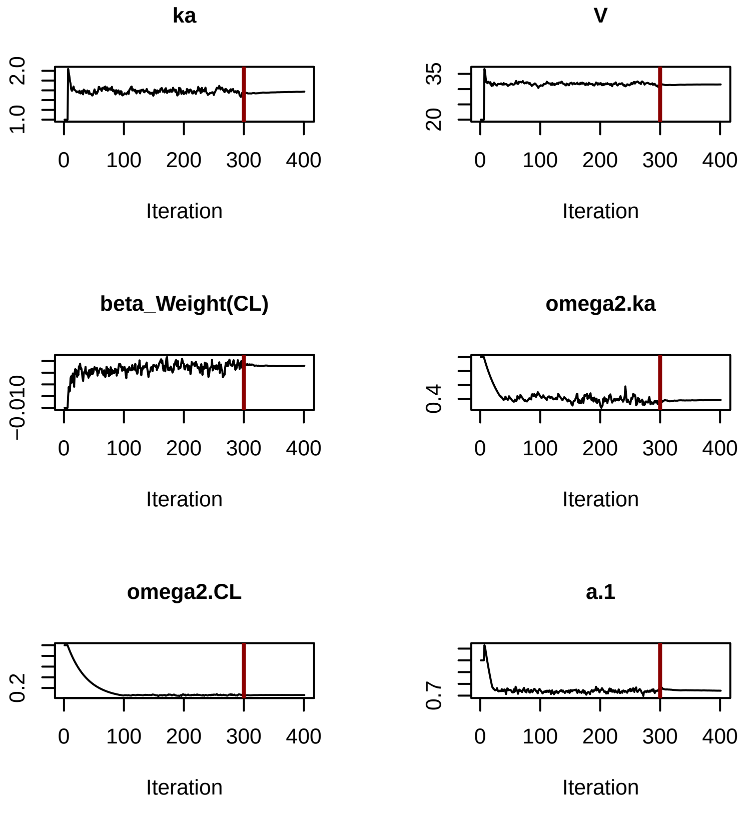
- **bug**
  - individual fit doesn't work (optim(par = phi1, fn = conditional.distribution_c, phii = phii, la fonction ne peut être évaluée aux paramètres initiaux) for theo.fit3 (check why)
  - covariate plots not working ("The following plot types were not found or are ambiguous: randeff.versus.covariates, parameters.versus.covariates")
- **TODO**
  - include new npde plots
  - include mirror plot
  - include diagnostic plots with samples from the conditional distribution (next version 3.1 ?)

```
myfit<-theo.fit1
# Generic plots
plot(myfit)
```

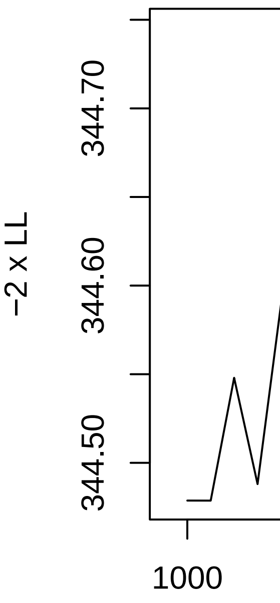
```
## Simulating data using nsim = 1000 simulated datasets
## Computing WRES and npde ..
```



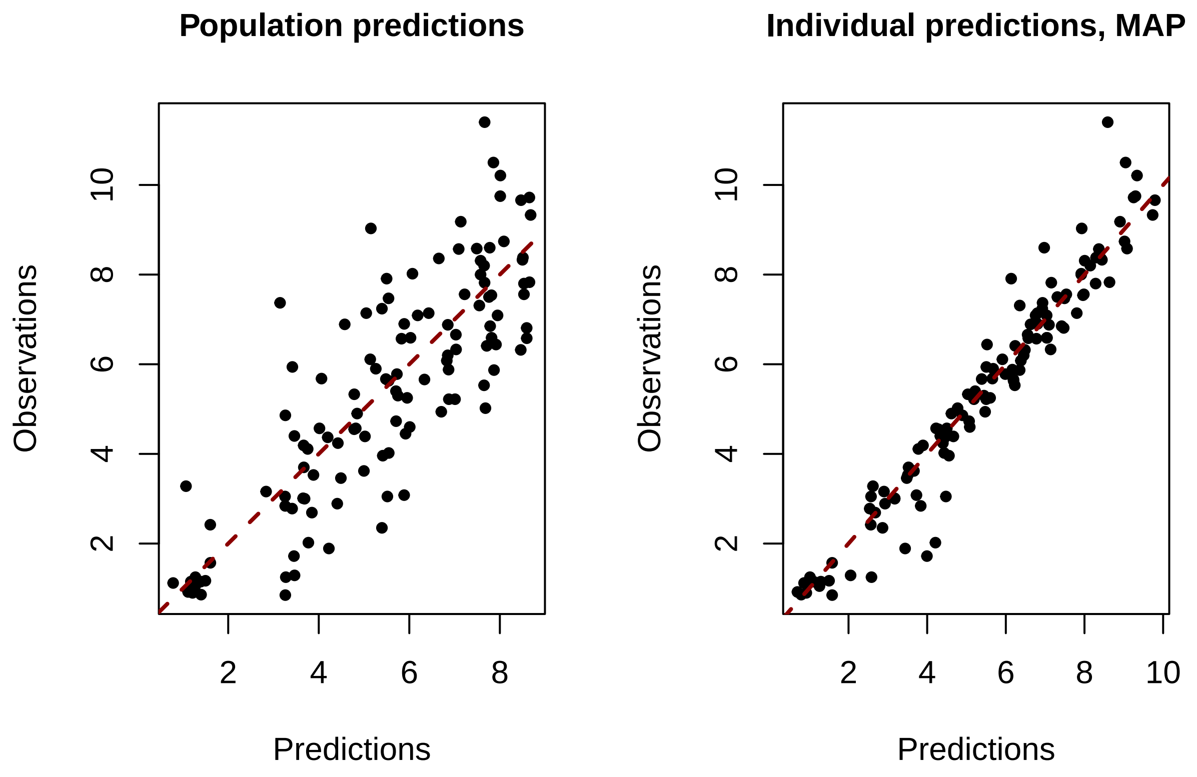
Concentration (mg/L)
Time (hr)



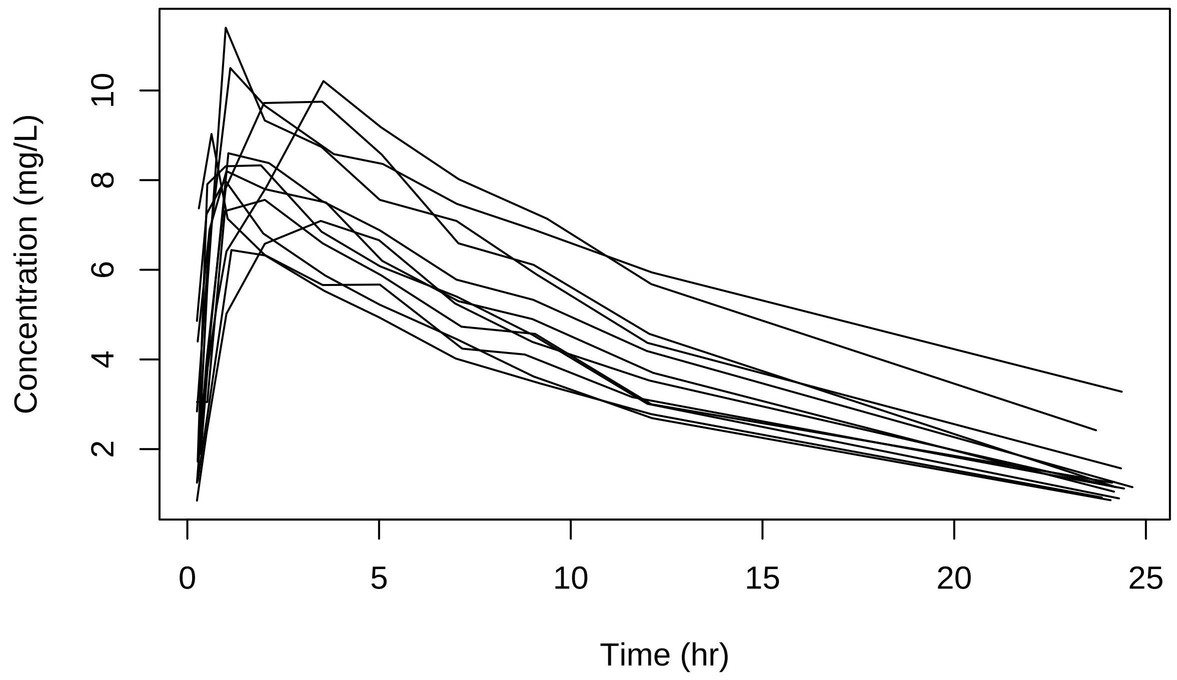
ka
V
CL
beta_Weight(CL)
omega2.ka
omega2.V
omega2.CL
a.1
Iteration



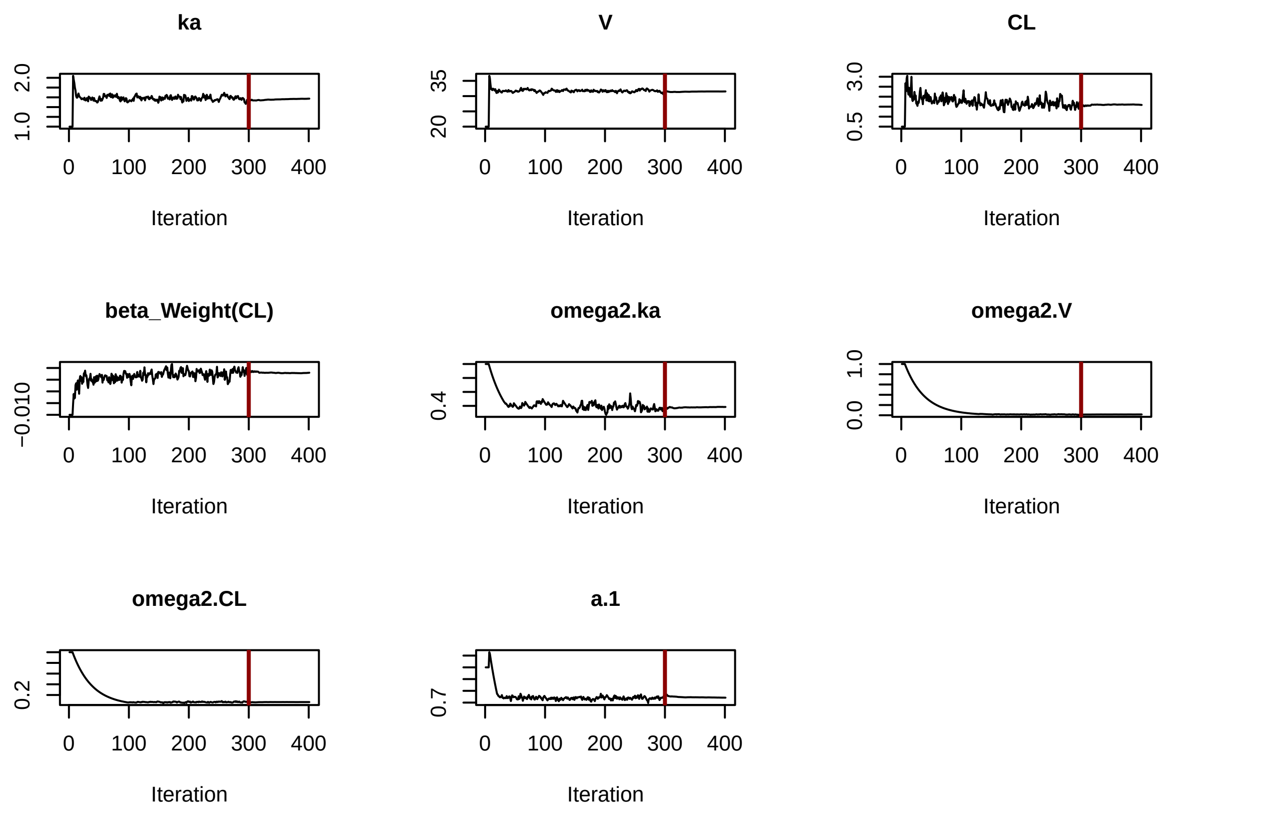
−2 x LL
344.70
344.60
344.50
1000



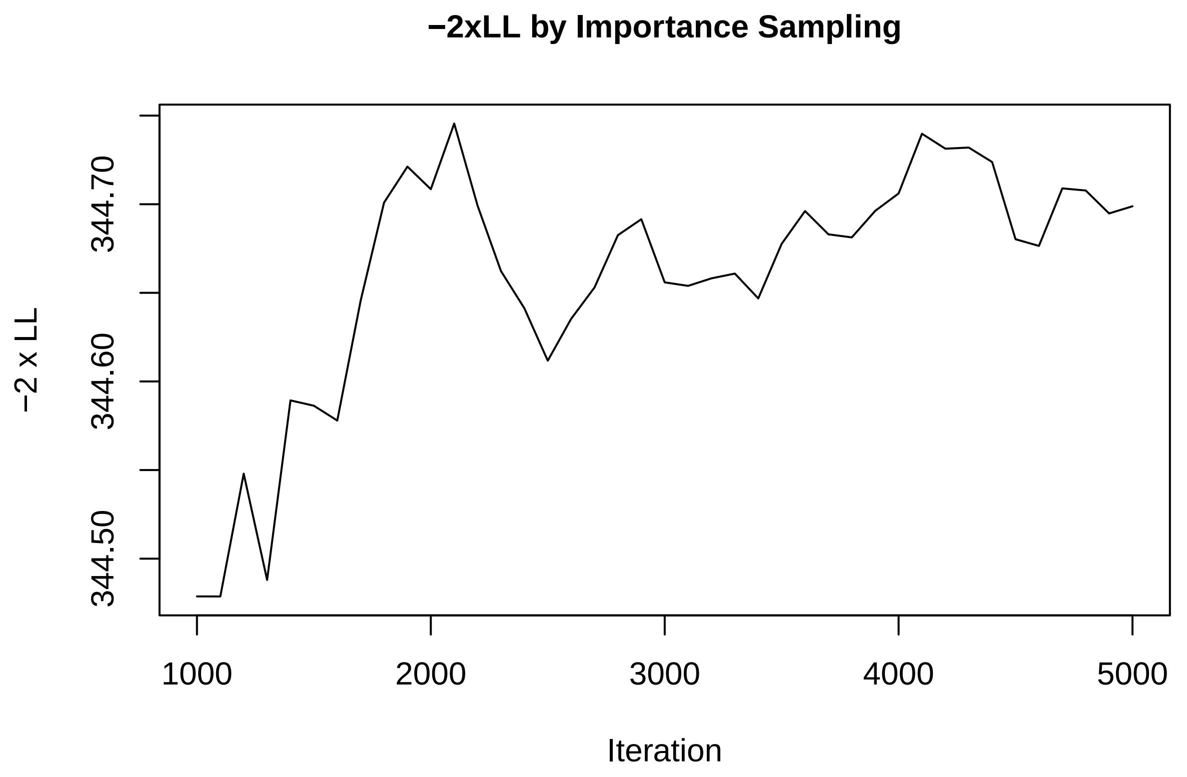
```r
# Individual plots
plot(myfit, plot.type="data")
```



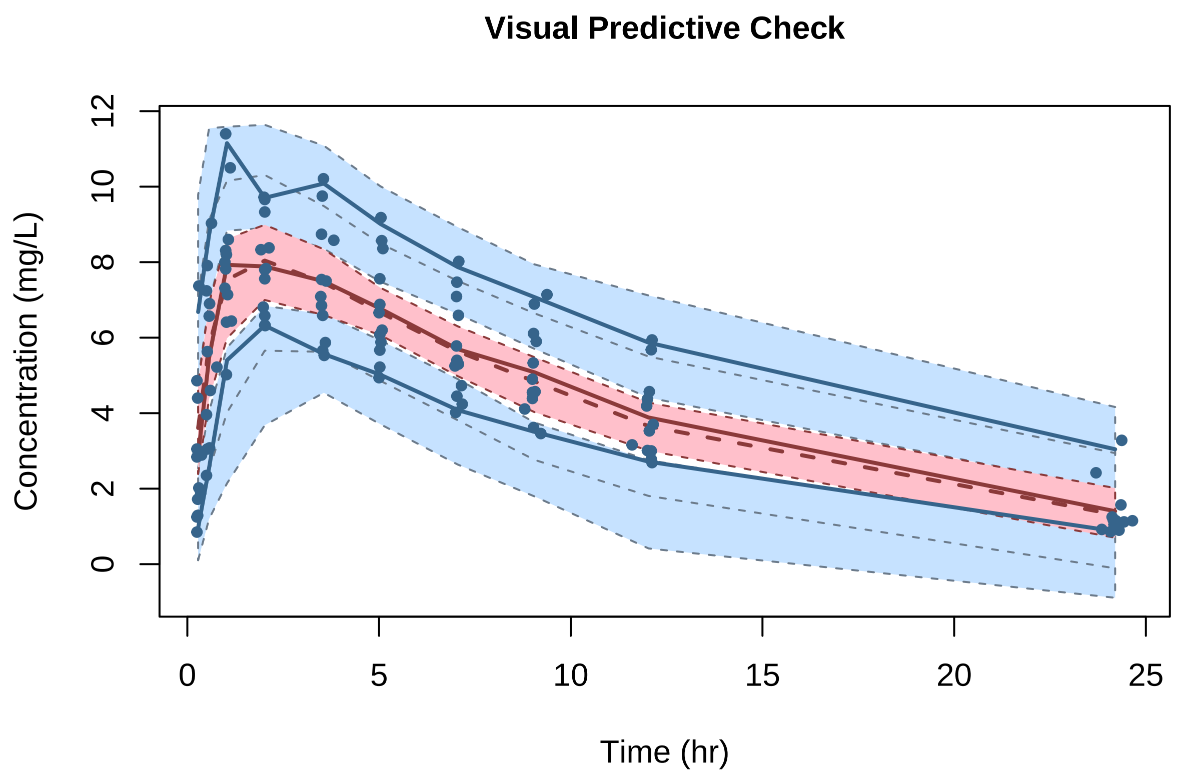
```r
plot(myfit, plot.type="convergence")
```



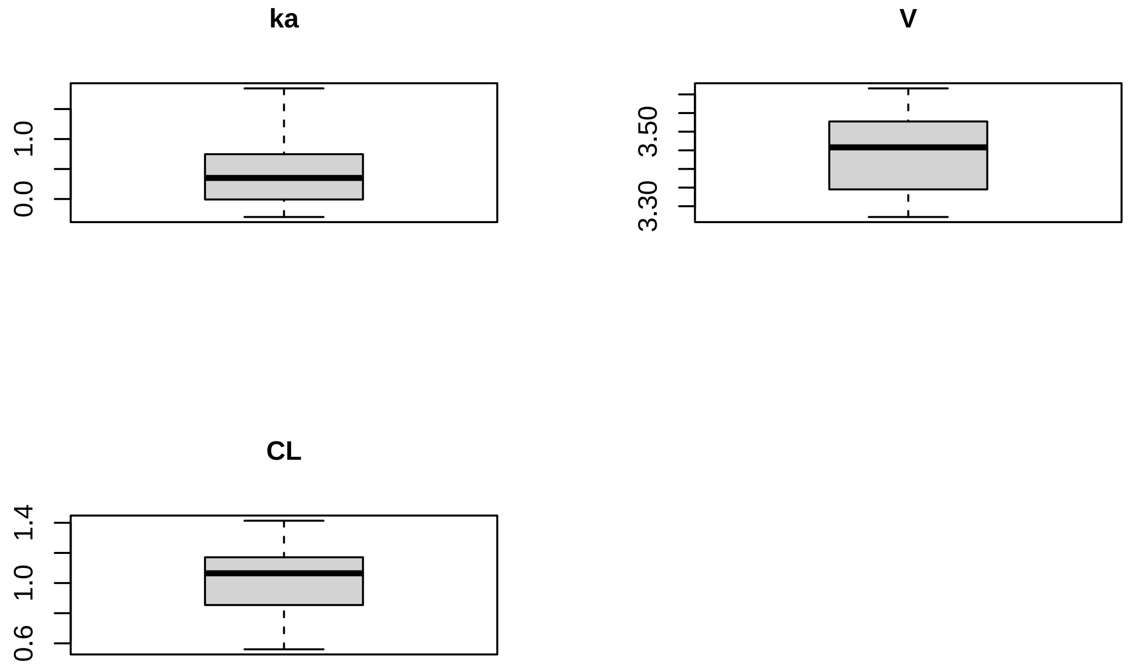
```r
plot(myfit, plot.type="likelihood")
```



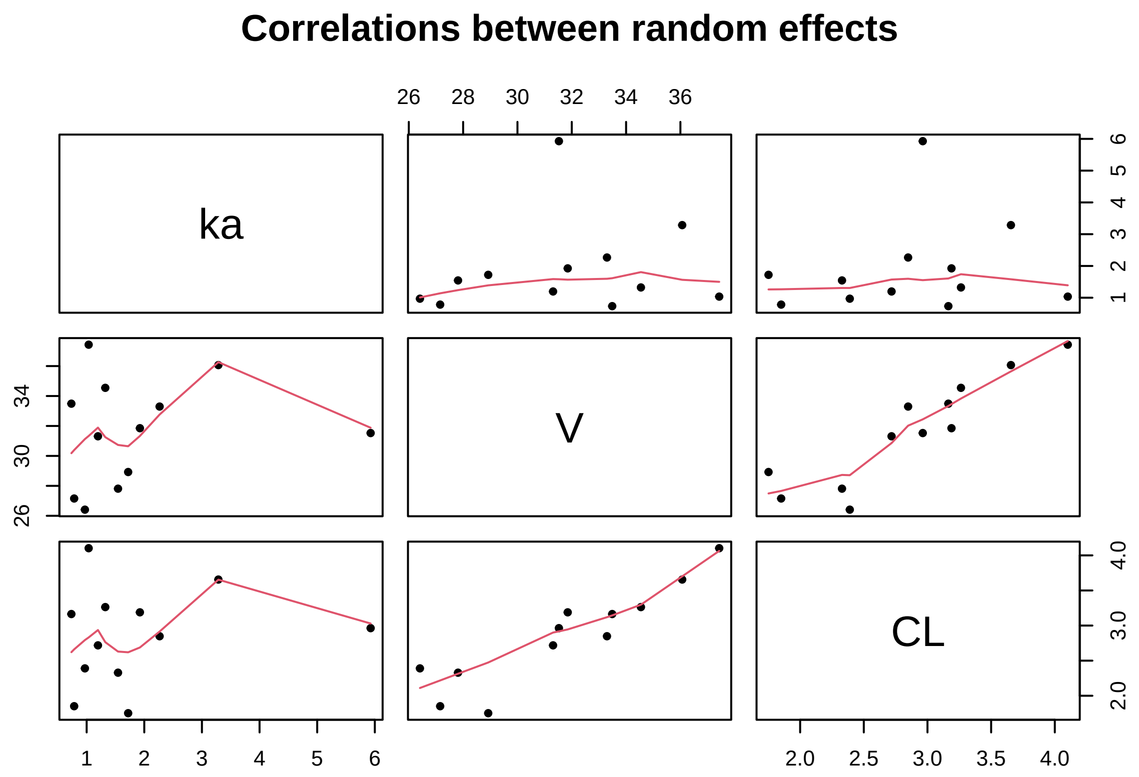
```r
plot(myfit, plot.type="vpc")
```



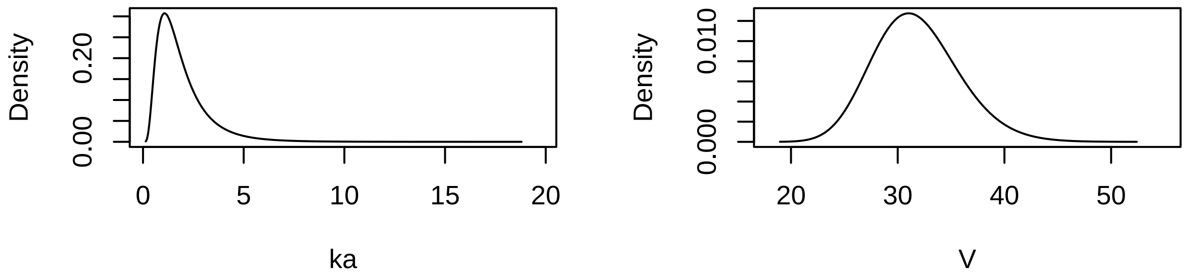
```
plot(myfit, plot.type="random.effects")
```

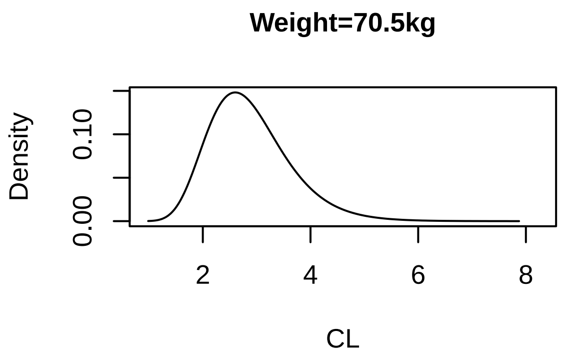


```
plot(myfit, plot.type="correlations")
```



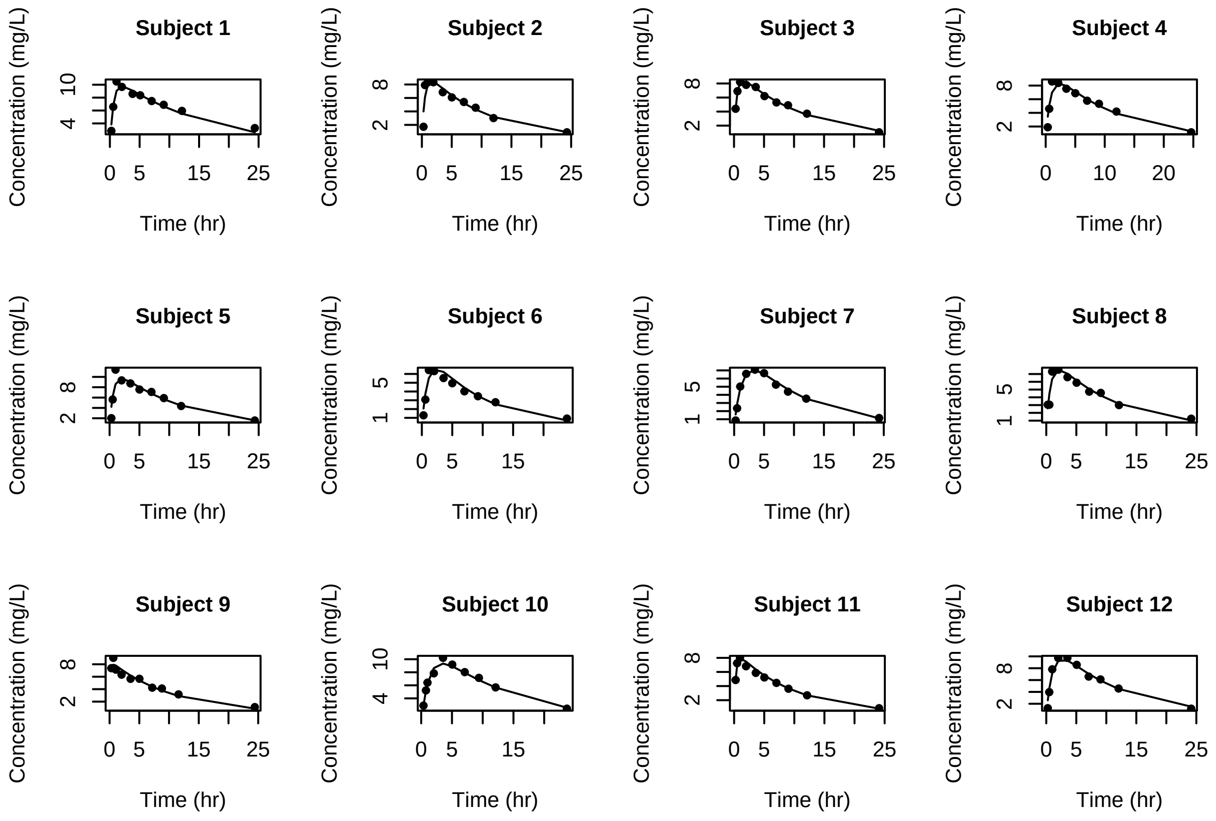
```r
plot(myfit, plot.type="marginal.distribution")
```





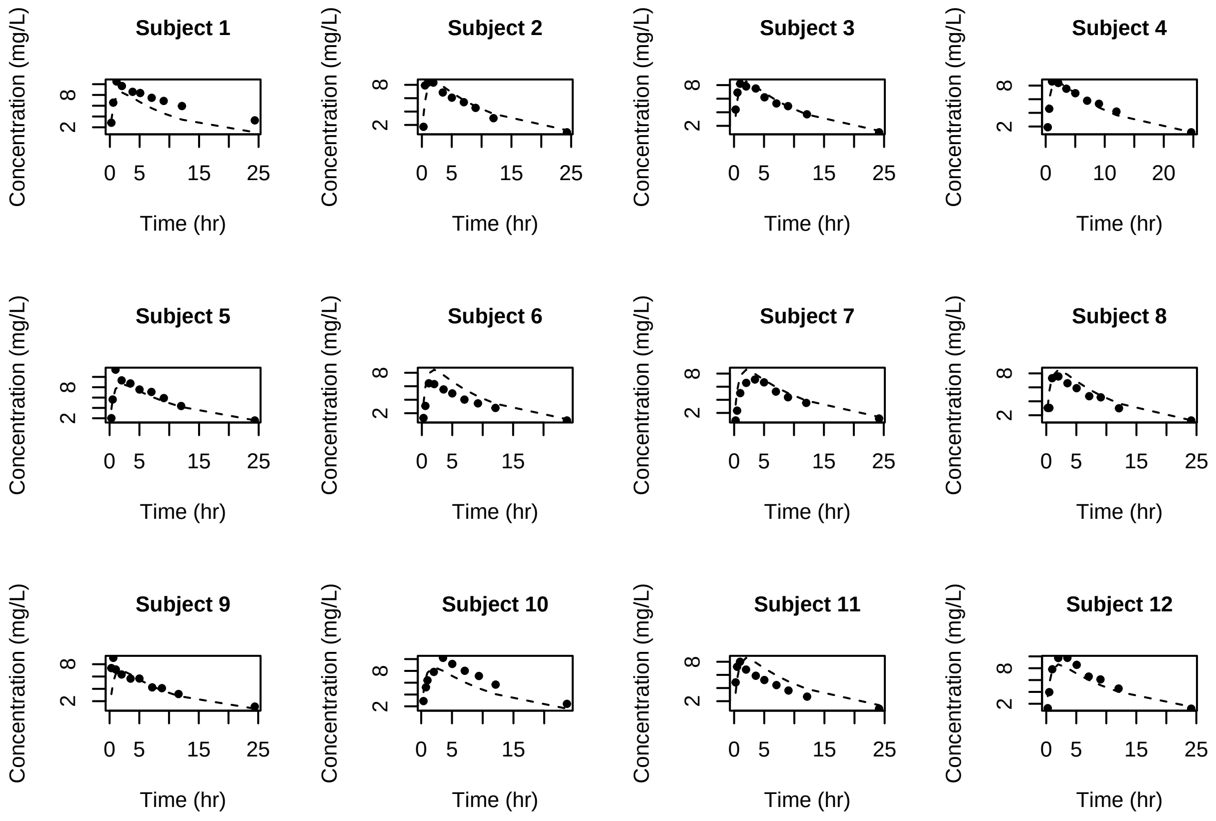
```r
plot(myfit, plot.type="individual.fit")
```

```
## Computing WRES and npde ..
```



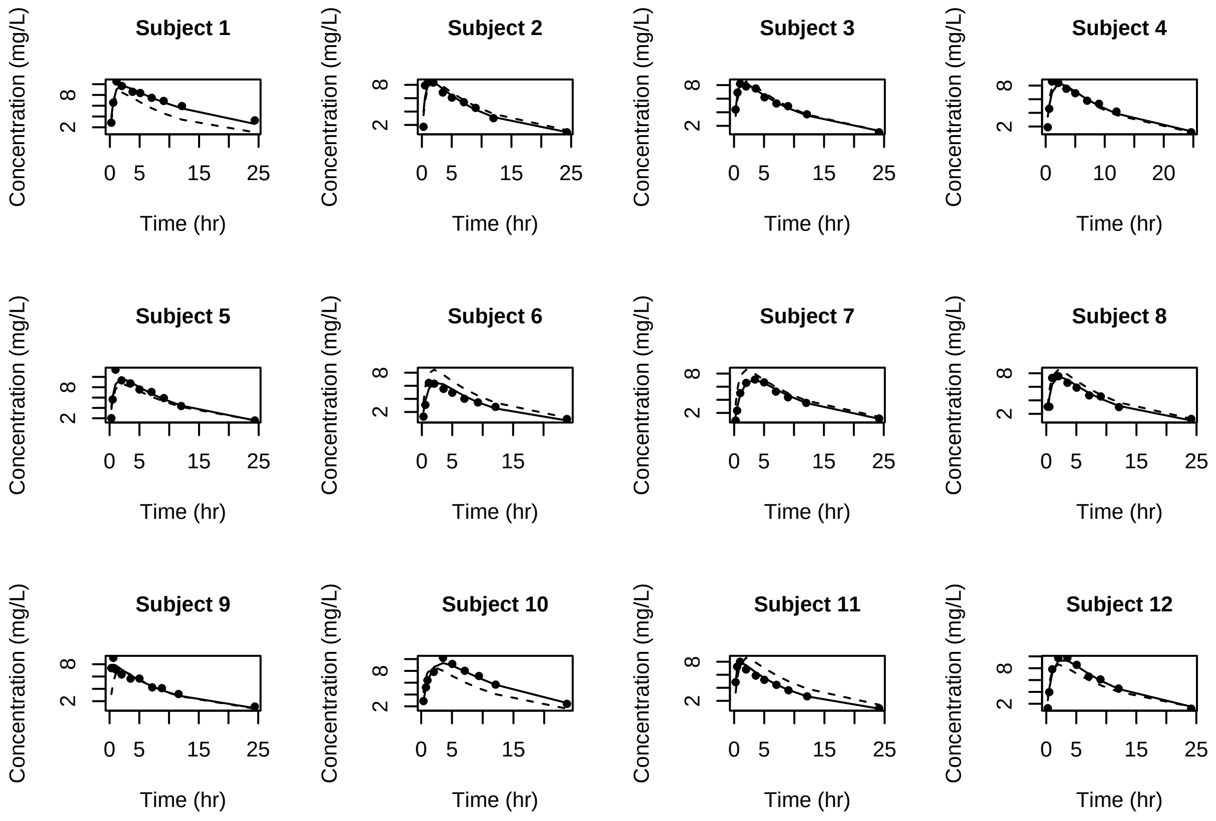
```
plot(myfit, plot.type="population.fit")
```

```
## Computing WRES and npde ..
```



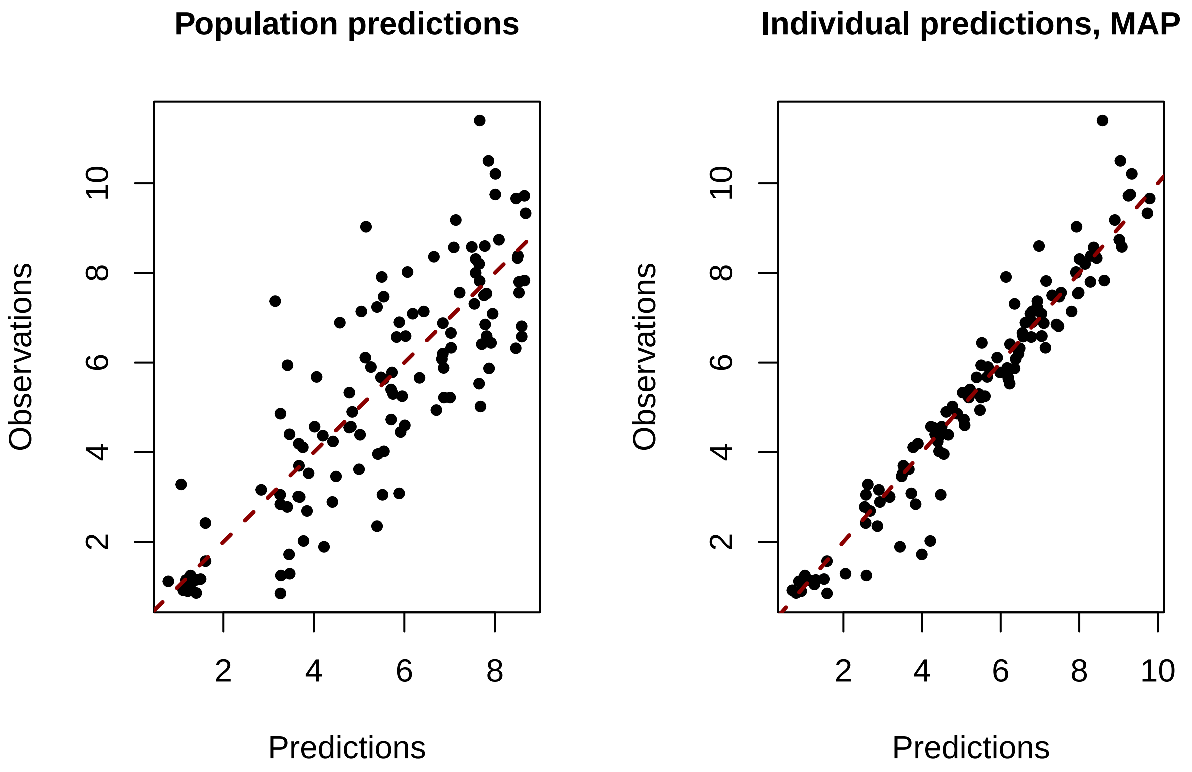
```r
plot(myfit, plot.type="both.fit")
```

```
## Computing WRES and npde ..
```

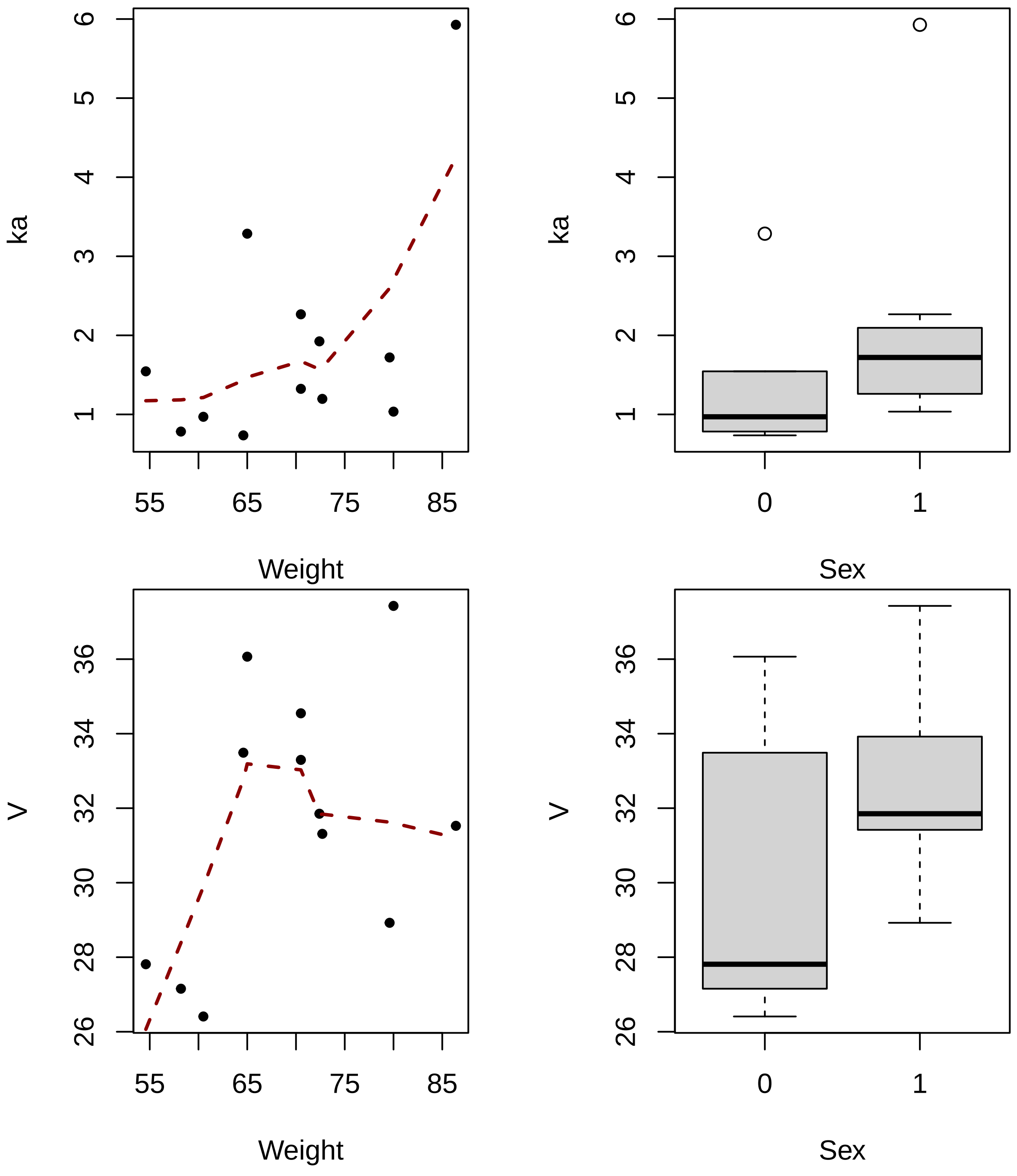


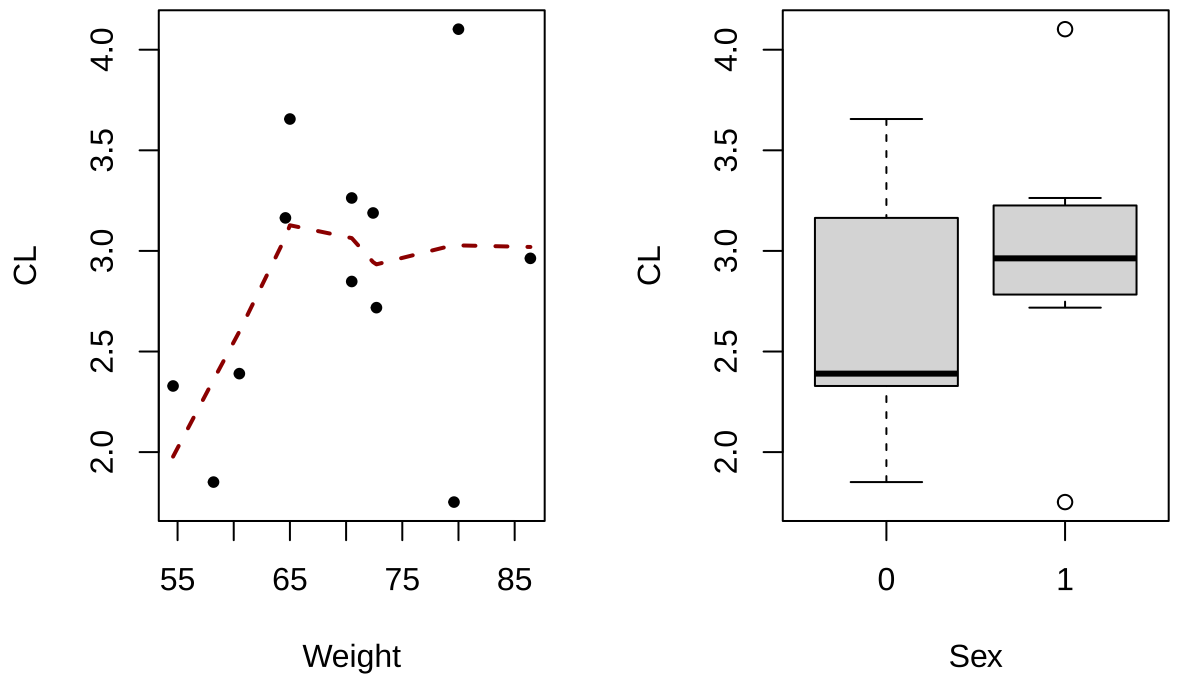
```r
plot(myfit, plot.type="observations.vs.predictions")
```

```
## Computing WRES and npde ..
```

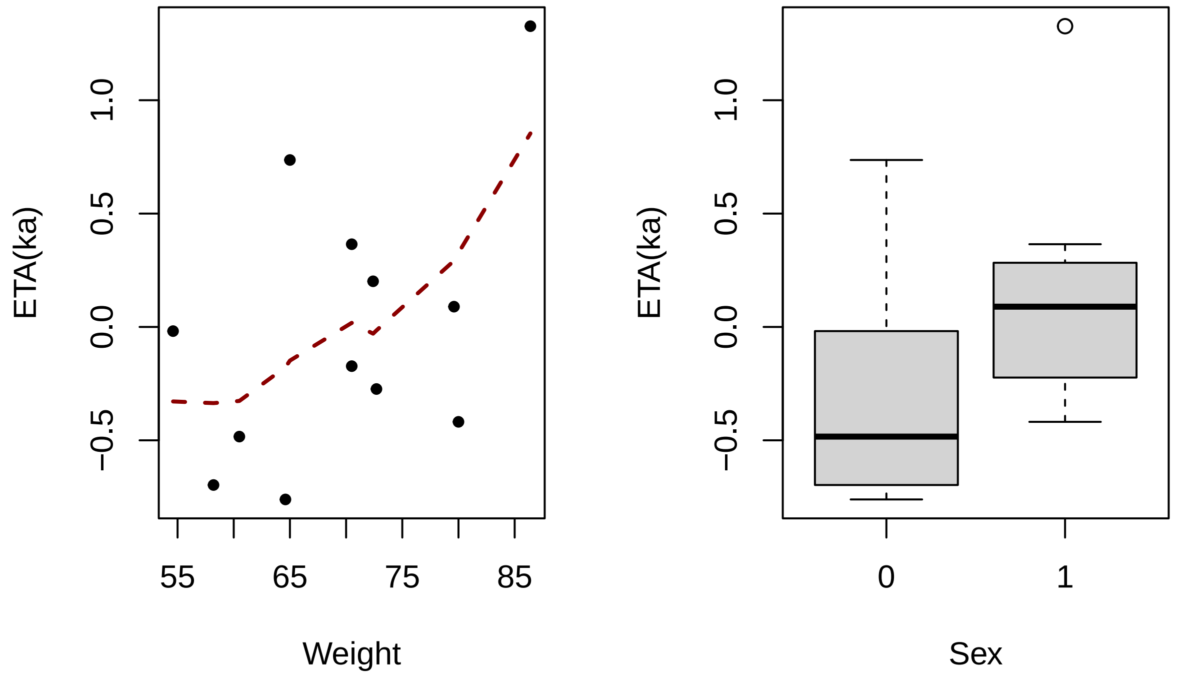


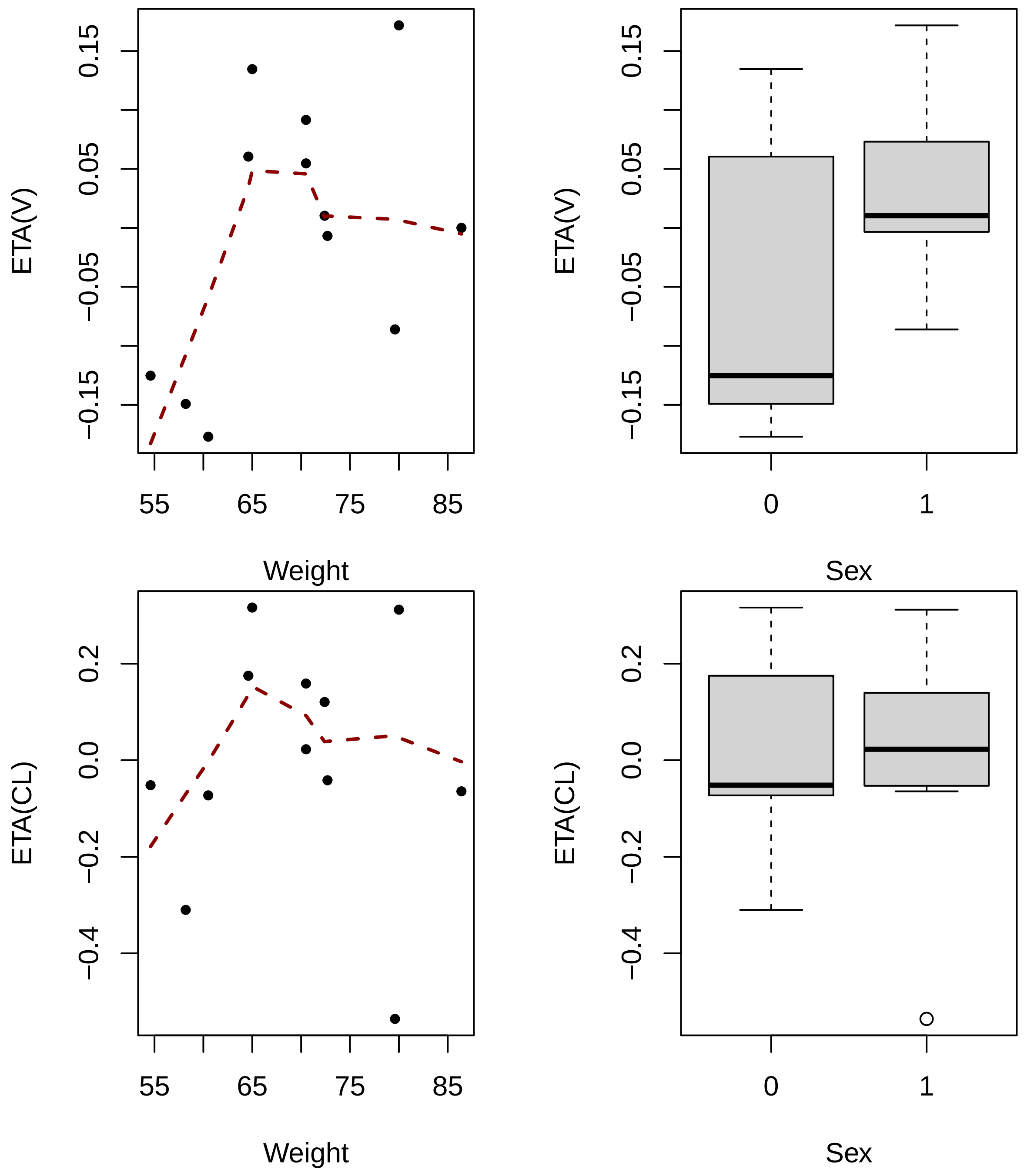
```r
# Not working
plot(myfit, plot.type="parameters.vs.covariates")
```





```
plot(myfit, plot.type="randeff.vs.covariates")
```



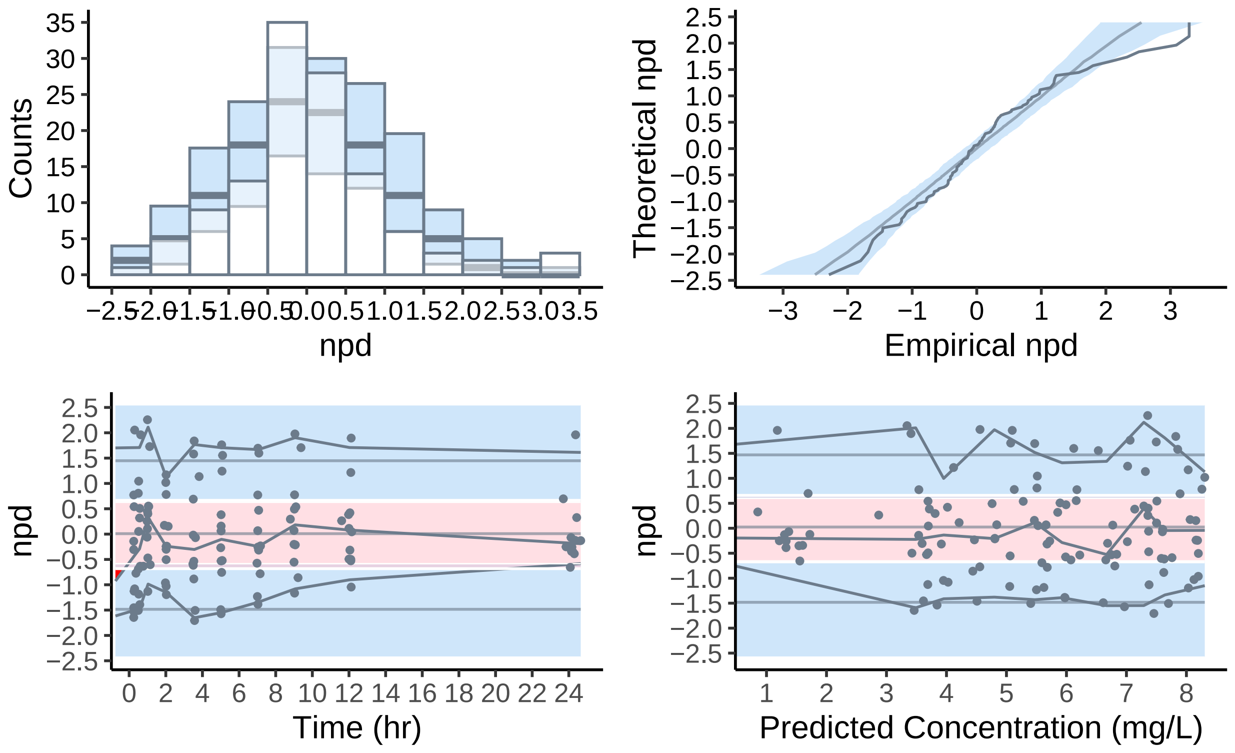


```r
# Warning
plot(myfit, plot.type="npde")
```

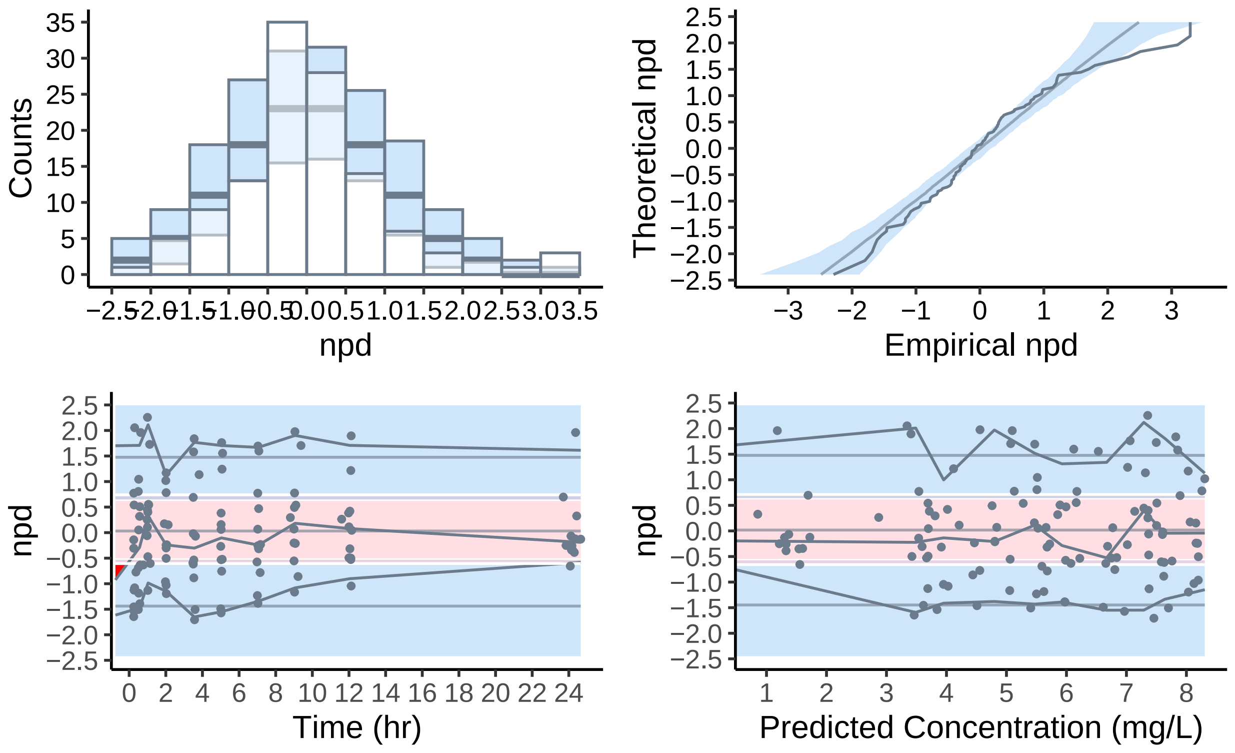
```
## Computing WRES and npde ..
## Please use npdeSaemix to obtain VPC and npde
```

```r
# Link to npde through npde library
ynpde <- npdeSaemix(myfit)
```

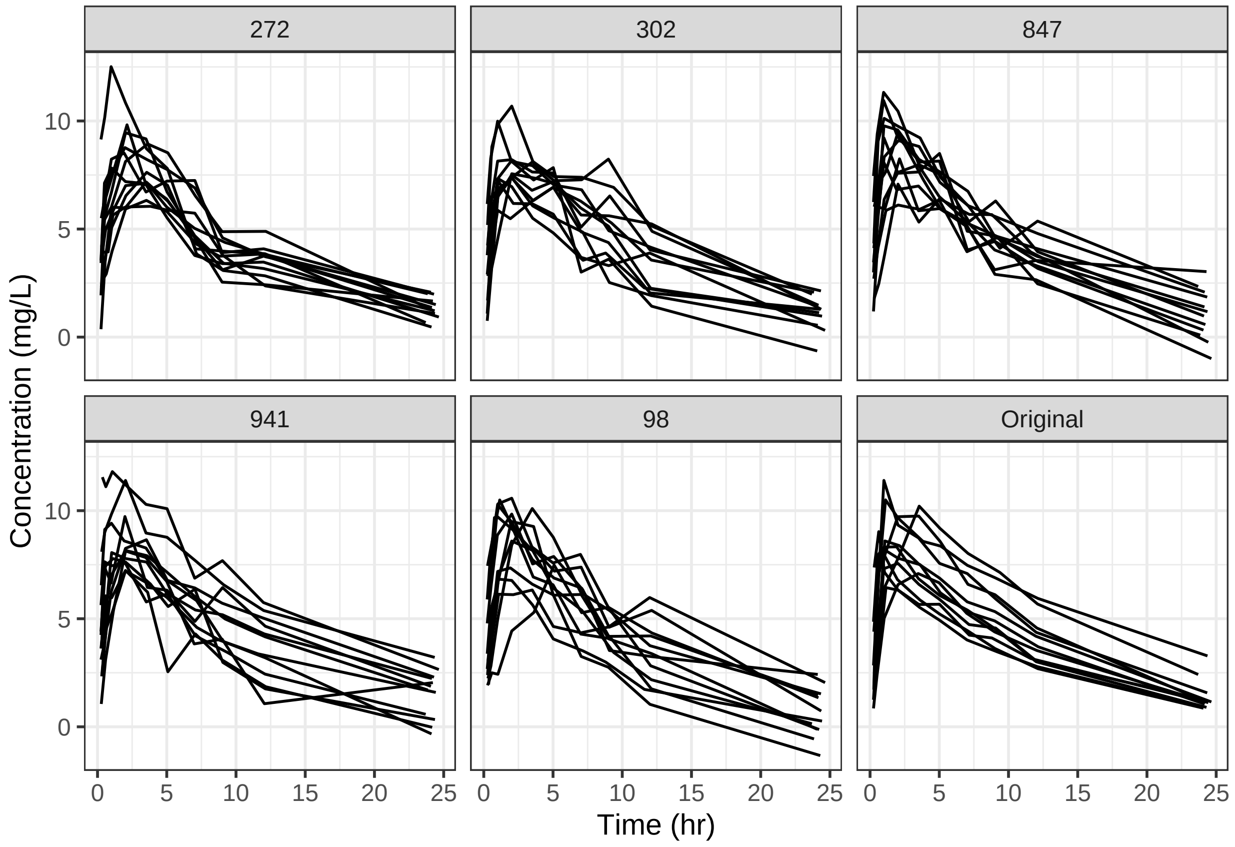
```
## ---------------------------------------------
## Distribution of npde :
##       nb of obs: 120
##            mean= 0.04905   (SE= 0.091 )
##        variance= 0.9839   (SE= 0.13 )
##        skewness= 0.828
##        kurtosis= 1.614
## ---------------------------------------------
## Statistical tests (adjusted p-values):
##   t-test                : 1
##   Fisher variance test  : 1
##   SW test of normality  : 0.000656 ***
##   Global test           : 0.000656 ***
## ---
## Signif. codes: '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
## ---------------------------------------------
```



```r
plot(ynpde)
```



```
# Simulations
mysim<-simulate.SaemixObject(myfit)

## Mirror plot
nmir<-5
isamp<-sample(1:mysim@sim.data@nsim, nmir, replace=FALSE)
datsim<-mysim@sim.data@datasim[mysim@sim.data@datasim$irep %in% isamp, ]
gdat<-cbind(mysim@data@data[,c(mysim@data@name.group, mysim@data@name.X, mysim@data@name.response)], da
gdat1<-data.frame(id=datsim[,c("idsim")], x=rep(gdat[,2],nmir),y=datsim[,"ysim"], data=as.character(dat
colnames(gdat)<-colnames(gdat1)
gdat<-rbind(gdat, gdat1)

ggplot(gdat, aes(x=x, y=y, group=id)) + geom_line() + facet_wrap(.~data, nrow=2, ncol=3) + theme_bw() +
```



Binary response

- **TODO**
  - error message “le nombre d’objets à remplacer n’est pas multiple de la taille du remplacement”
- **BUG**
  - check prediction function, the results don’t seem very good in terms of prediction of the data…
  - predict function should call predictions if not already there

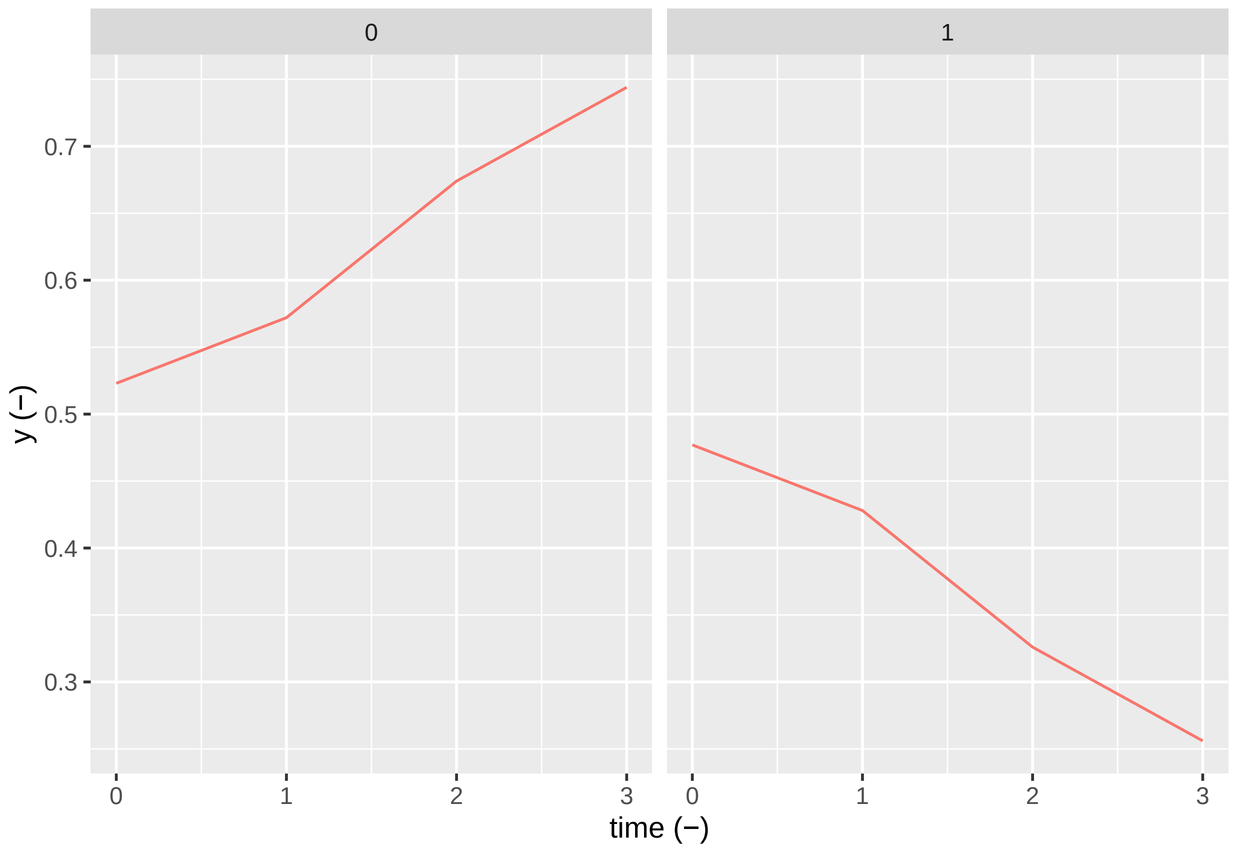
```
nsuj<-1000
xtim<-c(0:3)
parnam<-c("Intercept","beta.time")
param<-c(0,-0.37)
omega<-c(.21,.1)

partab<-as.data.frame(matrix(data=0,nrow=nsuj,ncol=2,dimnames=list(NULL,parnam)))
for(i in 1:2) partab[,i]<-rnorm(nsuj,mean=param[i],sd=omega[i])

psim<-data.frame()
for(itim in xtim) {
  logit.sim<-partab[,1]+partab[,2]*itim
  xtab<-exp(logit.sim)/(1+exp(logit.sim))
  psim<-rbind(psim,xtab)
}
datsim<-data.frame(id=rep(1:nsuj,each=length(xtim)),time=rep(xtim,nsuj),psim=unlist(psim))
rownames(datsim)<-NULL
ysim<-rbinom(nsuj*length(xtim),size=1,prob=datsim$psim)
summary(datsim)
```

```
##        id              time           psim        
##  Min.   :   1.0   Min.   :0.00   Min.   :0.09316  
##  1st Qu.: 250.8   1st Qu.:0.75   1st Qu.:0.29194  
##  Median : 500.5   Median :1.50   Median :0.37642  
##  Mean   : 500.5   Mean   :1.50   Mean   :0.37534  
##  3rd Qu.: 750.2   3rd Qu.:2.25   3rd Qu.:0.46229  
##  Max.   :1000.0   Max.   :3.00   Max.   :0.64514
```

```
datsim$y<-ysim
datsim$risk<-ifelse(datsim$id>500,1,0)

# Running saemix
saemix.data<-saemixData(name.data=datsim,
      name.group=c("id"),name.predictors=c("time","y"), name.covariates=c("risk"),name.X=c("time"))
```

```
## Column name(s)  do(es) not exist in the dataset, please check
## Remove columns 1 (  )
## No valid name given, attempting automatic recognition
## Automatic recognition of columns y successful
## 
## 
## The following SaemixData object was successfully created:
## 
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset datsim 
##     Structured data: y ~ time + y | id 
##     X variable for graphs: time () 
##     covariates: risk (-) 
##       reference class for covariate risk :  0
```

```
plotDiscreteData(saemix.data, outcome='binary')
```



```r
binary.model<-function(psi,id,xidep) {
  tim<-xidep[,1]
  y<-xidep[,2]
  inter<-psi[id,1]
  slope<-psi[id,2]
  logit<-inter+slope*tim
  pevent<-exp(logit)/(1+exp(logit))
  logpdf<-rep(0,length(tim))
  P.obs = (y==0)*(1-pevent)+(y==1)*pevent
  logpdf <- log(P.obs)
  return(logpdf)
}

simulBinary<-function(psi,id,xidep) {
  tim<-xidep[,1]
  y<-xidep[,2]
  inter<-psi[id,1]
  slope<-psi[id,2]
  logit<-inter+slope*tim
  pevent<-exp(logit)/(1+exp(logit))
  ysim<-rbinom(length(tim),size=1, prob=pevent)
  return(ysim)
}

saemix.model<-saemixModel(model=binary.model,description="Binary model",
        modeltype="likelihood", simulate.function=simulBinary,
        psi0=matrix(c(0,-.5,0.5,0),ncol=2,byrow=TRUE,dimnames=list(NULL,parnam[1:2])),
        transform.par=c(0,0),covariance.model=matrix(c(1,0,0,1),ncol=2))
```

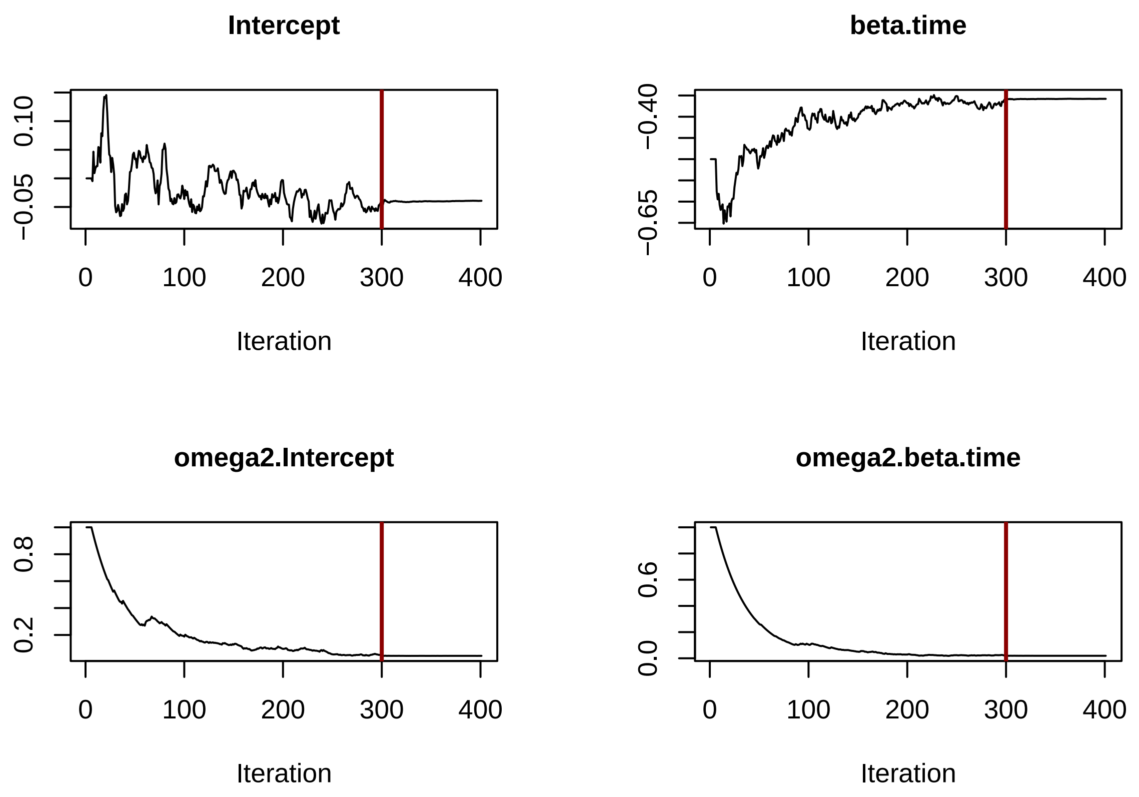
```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  Binary model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   tim<-xidep[,1]
##   y<-xidep[,2]
##   inter<-psi[id,1]
##   slope<-psi[id,2]
##   logit<-inter+slope*tim
##   pevent<-exp(logit)/(1+exp(logit))
##   logpdf<-rep(0,length(tim))
##   P.obs = (y==0)*(1-pevent)+(y==1)*pevent
##   logpdf <- log(P.obs)
##   return(logpdf)
## }
##   Nb of parameters: 2
##       parameter names:  Intercept beta.time
##       distribution:
##      Parameter Distribution Estimated
## [1,] Intercept normal       Estimated
## [2,] beta.time normal       Estimated
##   Variance-covariance matrix:
##           Intercept beta.time
## Intercept         1         0
## beta.time         0         1
##     No covariate in the model.
##     Initial values
##              Intercept beta.time
## Pop.CondInit       0.0      -0.5
## Cov.CondInit       0.5       0.0
```

```
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, fim=FALSE, displayProgress=FALSE)
# saemix.fit<-saemix(saemix.model,saemix.data,saemix.options)

binary.fit<-saemix(saemix.model,saemix.data,saemix.options)
```

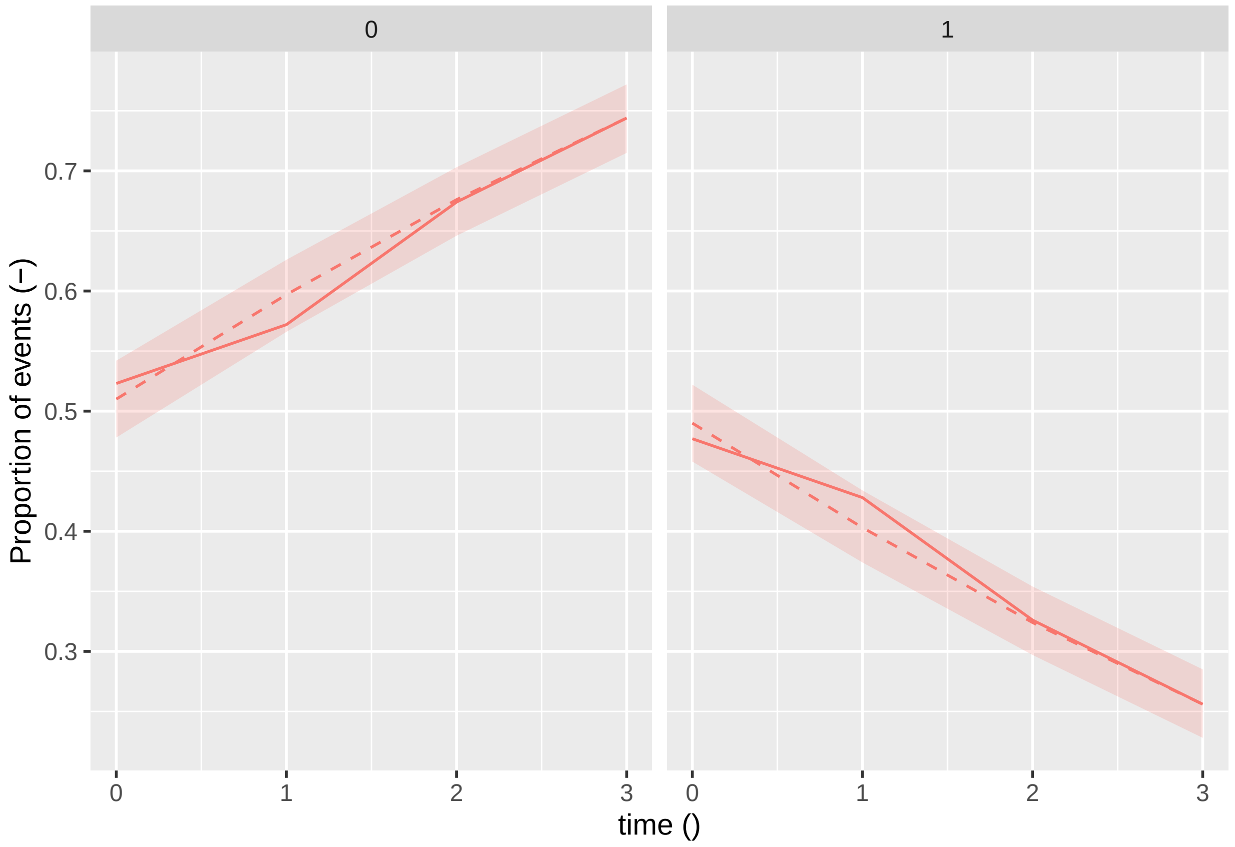
```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data             ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset datsim
##     Structured data: y ~ time + y | id
##     X variable for graphs: time ()
##     covariates: risk (-)
##       reference class for covariate risk :  0
## Dataset characteristics:
```

```
##     number of subjects:     1000
##     number of observations: 4000
##     average/min/max nb obs: 4.00  /  4  /  4
## First 10 lines of data:
##    id time y y.1 risk mdv cens occ ytype
## 1   1    0 1   1    0   0    0   1     1
## 2   1    1 0   0    0   0    0   1     1
## 3   1    2 1   1    0   0    0   1     1
## 4   1    3 0   0    0   0    0   1     1
## 5   2    0 0   0    0   0    0   1     1
## 6   2    1 1   1    0   0    0   1     1
## 7   2    2 0   0    0   0    0   1     1
## 8   2    3 0   0    0   0    0   1     1
## 9   3    0 0   0    0   0    0   1     1
## 10  3    1 1   1    0   0    0   1     1
## ----------------------------------
## ----          Model           ----
## ----------------------------------
## Nonlinear mixed-effects model
##   Model function:  Binary model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   tim<-xidep[,1]
##   y<-xidep[,2]
##   inter<-psi[id,1]
##   slope<-psi[id,2]
##   logit<-inter+slope*tim
##   pevent<-exp(logit)/(1+exp(logit))
##   logpdf<-rep(0,length(tim))
##   P.obs = (y==0)*(1-pevent)+(y==1)*pevent
##   logpdf <- log(P.obs)
##   return(logpdf)
## }
## <bytecode: 0x55eb2271daf8>
##   Nb of parameters: 2
##       parameter names:  Intercept beta.time
##       distribution:
##      Parameter Distribution Estimated
## [1,] Intercept normal       Estimated
## [2,] beta.time normal       Estimated
##   Variance-covariance matrix:
##           Intercept beta.time
## Intercept         1         0
## beta.time         0         1
##     No covariate in the model.
##     Initial values
##              Intercept beta.time
## Pop.CondInit         0      -0.5
## ----------------------------------
## ----    Key algorithm options  ----
## ----------------------------------
##     Estimation of individual parameters (MAP)
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=300, K2=100
##     Number of chains:  1
##     Seed:  632545
##     Number of MCMC iterations for IS:  5000
##     Simulations:
##         nb of simulated datasets used for npde:  1000
##         nb of simulated datasets used for VPC:  100
##     Input/output
##         save the results to a file:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate
## [1,] Intercept -0.039
## [2,] beta.time -0.358
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##           Parameter        Estimate
## Intercept omega2.Intercept 0.045
## beta.time omega2.beta.time 0.019
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##                  omega2.Intercept omega2.beta.time
## omega2.Intercept 1                0
## omega2.beta.time 0                1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
##
## Likelihood computed by importance sampling
##       -2LL= 5155.427
##       AIC = 5165.427
##       BIC = 5189.966
## ----------------------------------------------------
```

```
plot(binary.fit, plot.type="convergence")
```



```
nsim<-1000
binary.fit2 <- simulateDiscreteSaemix(binary.fit, nsim=nsim)
if(FALSE){
  object<-binary.fit
}

discreteVPC(binary.fit2, outcome="binary")
```



```
vec<-predict(binary.fit)
ypred<-as.numeric(exp(vec)>=0.5)
print(table(ypred,datsim$y))
```

```
##
## ypred    0    1
##     0   13 1362
##     1 2500  125
```

```
print(fisher.test(table(ypred,datsim$y)))
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  table(ypred, datsim$y)
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.0002325439 0.0008300781
## sample estimates:
##   odds ratio
## 0.0004880112
```

Ordinal data

Ordinal example from Belhal

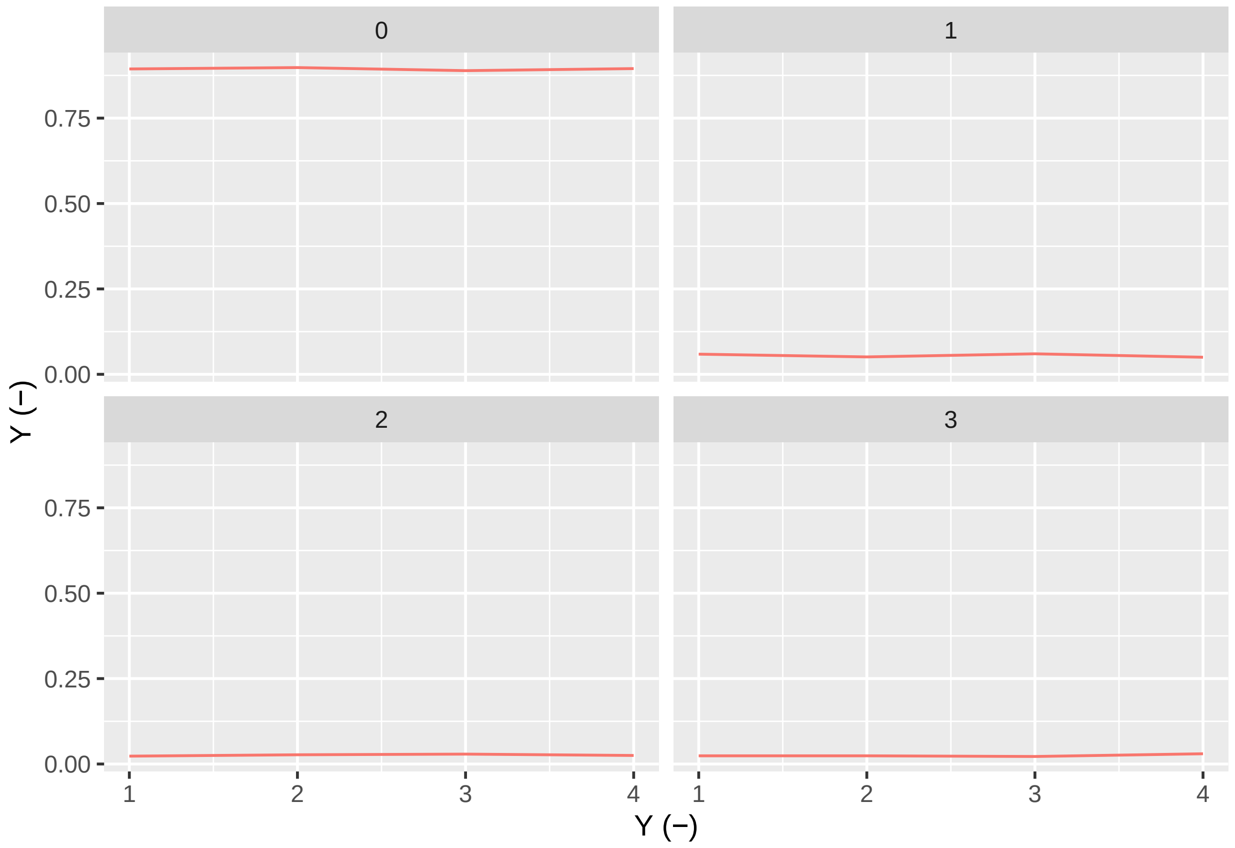
TODO: check the example step by step, no time effect ?

- **TODO**
  - check results compared to previous version
  - check LL by GQ to compare to LL by IS
  - test the optimisation and change the algorithm for one-dimension

```
smx.ord <- read.table(file.path(belDir,"categorical1_data.txt"),header=T)
saemix.data<-saemixData(name.data=smx.ord, header=TRUE, sep=" ", na=NA,
                        name.group=c("ID"), name.predictors=c("Y","TIME"), name.X=c("TIME"))
```

```
## Column name(s)  do(es) not exist in the dataset, please check
## Remove columns 1 (  )
## No valid name given, attempting automatic recognition
## Automatic recognition of columns Y successful
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset smx.ord
##     Structured data: Y ~ Y + TIME | ID
##     X variable for graphs: TIME ()
```

```
plotDiscreteData(saemix.data, outcome='categorical')
```



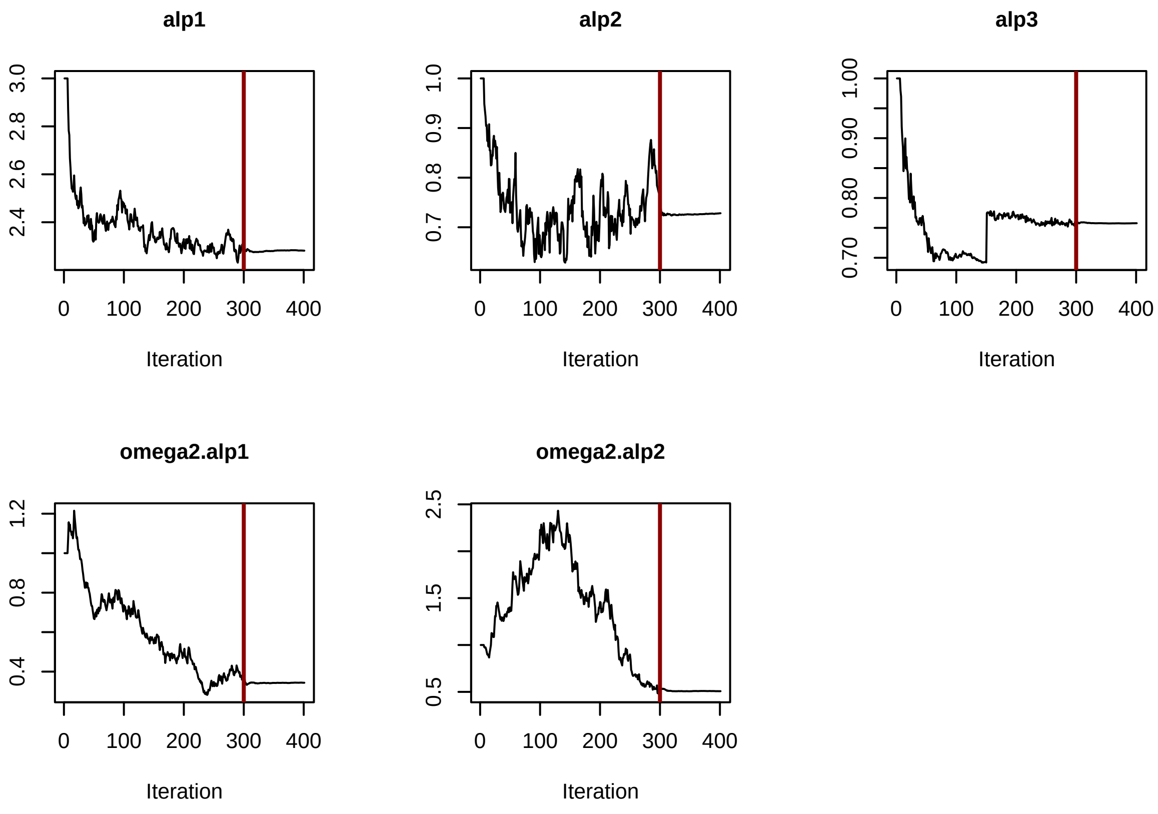
```
# Model
ordinal.model<-function(psi,id,xidep) {
  y<-xidep[,1]
  alp1<-psi[id,1]
  alp2<-psi[id,2]
  alp3<-psi[id,3]
  logit1<-alp1
  logit2<-alp1+alp2
  logit3<-alp1+alp2+alp3
  pge1<-exp(logit1)/(1+exp(logit1))
  pge2<-exp(logit2)/(1+exp(logit2))
  pge3<-exp(logit3)/(1+exp(logit3))
  logpdf<-rep(0,length(y))
  P.obs = (y==0)*pge1+(y==1)*(pge2 - pge1)+(y==2)*(pge3 - pge2)+(y==3)*(1 - pge3)
  logpdf <- log(P.obs)
  return(logpdf)
}

simulateOrdinal<-function(psi,id,xidep) {
  y<-xidep[,1]
  alp1<-psi[id,1]
  alp2<-psi[id,2]
  alp3<-psi[id,3]
  logit1<-alp1
  logit2<-alp1+alp2
  logit3<-alp1+alp2+alp3
  pge1<-exp(logit1)/(1+exp(logit1))
  pge2<-exp(logit2)/(1+exp(logit2))
  pge3<-exp(logit3)/(1+exp(logit3))
  x<-runif(length(y))
  ysim<-1+as.integer(x>pge1)+as.integer(x>pge2)+as.integer(x>pge3)
  return(ysim)
}

saemix.model<-saemixModel(model=ordinal.model,description="Ordinal categorical model",modeltype="likeli
                          simulate.function=simulateOrdinal,
                          psi0=matrix(c(3,1,1),ncol=3,byrow=TRUE,dimnames=list(NULL,c("alp1","alp2","al
                          omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
                          transform.par=c(0,1,1),covariance.model=matrix(c(1,0,0,0,1,0,0,0,0),ncol=3))
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  Ordinal categorical model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   y<-xidep[,1]
##   alp1<-psi[id,1]
##   alp2<-psi[id,2]
##   alp3<-psi[id,3]
##   logit1<-alp1
##   logit2<-alp1+alp2
##   logit3<-alp1+alp2+alp3
##   pge1<-exp(logit1)/(1+exp(logit1))
##   pge2<-exp(logit2)/(1+exp(logit2))
##   pge3<-exp(logit3)/(1+exp(logit3))
##   logpdf<-rep(0,length(y))
##   P.obs = (y==0)*pge1+(y==1)*(pge2 - pge1)+(y==2)*(pge3 - pge2)+(y==3)*(1 - pge3)
##   logpdf <- log(P.obs)
##   return(logpdf)
## }
##   Nb of parameters: 3
##       parameter names:  alp1 alp2 alp3
##       distribution:
##      Parameter Distribution Estimated
## [1,] alp1      normal       Estimated
## [2,] alp2      log-normal   Estimated
## [3,] alp3      log-normal   Estimated
##   Variance-covariance matrix:
##      alp1 alp2 alp3
## alp1    1    0    0
## alp2    0    1    0
## alp3    0    0    0
##     No covariate in the model.
##     Initial values
##              alp1 alp2 alp3
## Pop.CondInit    3    1    1
```

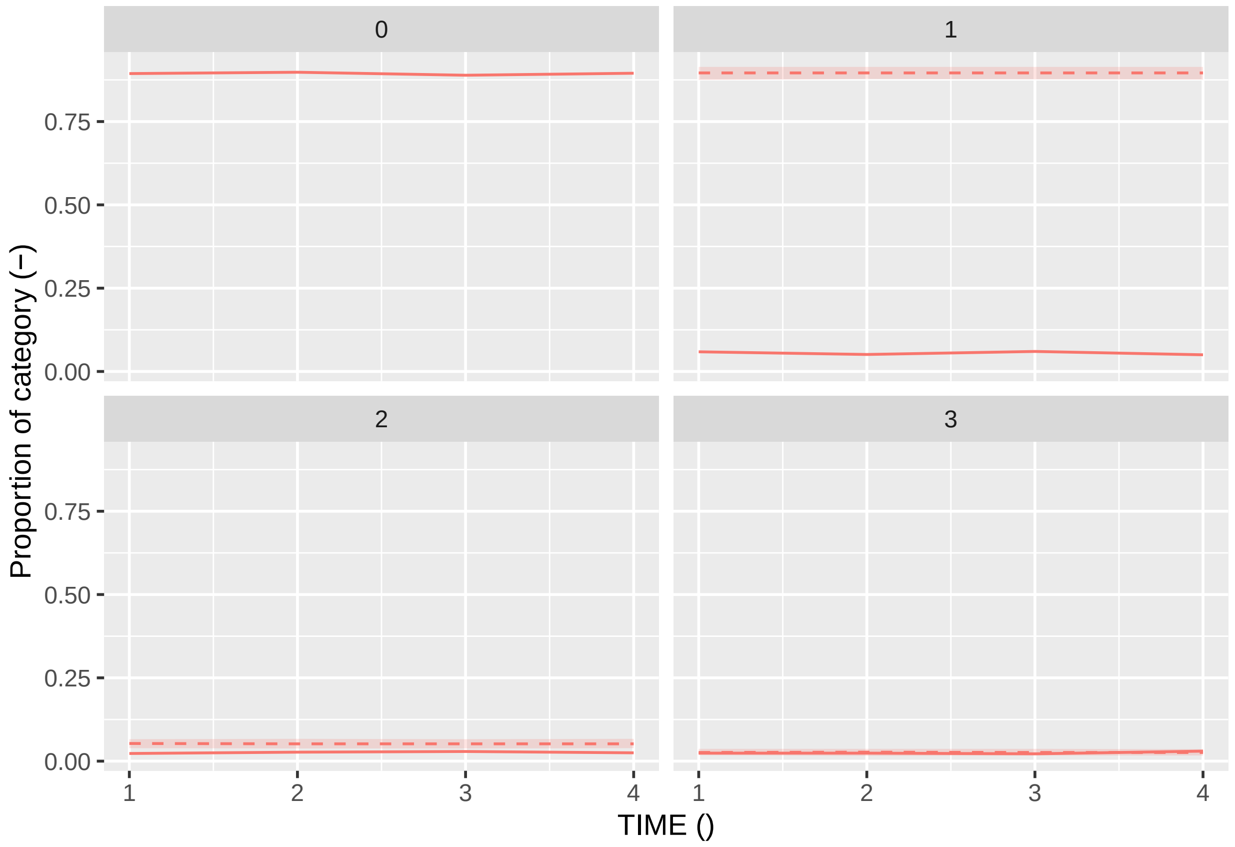
```
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, fim=FALSE, displayProgress=FALSE)
saemix.fit<-saemix(saemix.model,saemix.data,saemix.options)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data              ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset smx.ord
##     Structured data: Y ~ Y + TIME | ID
##     X variable for graphs: TIME ()
## Dataset characteristics:
##     number of subjects:     1000
##     number of observations: 4000
##     average/min/max nb obs: 4.00  /  4  /  4
## First 10 lines of data:
##    ID Y TIME Y.1 mdv cens occ ytype
## 1   1 3    1   3   0    0   1     1
## 2   1 0    2   0   0    0   1     1
## 3   1 0    3   0   0    0   1     1
## 4   1 0    4   0   0    0   1     1
## 5   2 3    1   3   0    0   1     1
## 6   2 0    2   0   0    0   1     1
## 7   2 0    3   0   0    0   1     1
## 8   2 0    4   0   0    0   1     1
## 9   3 0    1   0   0    0   1     1
## 10  3 0    2   0   0    0   1     1
## -----------------------------------
## ----          Model             ----
## -----------------------------------
## Nonlinear mixed-effects model
##   Model function:  Ordinal categorical model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   y<-xidep[,1]
##   alp1<-psi[id,1]
##   alp2<-psi[id,2]
##   alp3<-psi[id,3]
##   logit1<-alp1
##   logit2<-alp1+alp2
##   logit3<-alp1+alp2+alp3
##   pge1<-exp(logit1)/(1+exp(logit1))
##   pge2<-exp(logit2)/(1+exp(logit2))
##   pge3<-exp(logit3)/(1+exp(logit3))
##   logpdf<-rep(0,length(y))
##   P.obs = (y==0)*pge1+(y==1)*(pge2 - pge1)+(y==2)*(pge3 - pge2)+(y==3)*(1 - pge3)
##   logpdf <- log(P.obs)
##   return(logpdf)
## }
## <bytecode: 0x55eb1d69c730>
##   Nb of parameters: 3
##       parameter names:  alp1 alp2 alp3
##       distribution:
##      Parameter Distribution Estimated
## [1,] alp1      normal       Estimated
## [2,] alp2      log-normal   Estimated
## [3,] alp3      log-normal   Estimated
##   Variance-covariance matrix:
##      alp1 alp2 alp3
## alp1    1    0    0
## alp2    0    1    0
## alp3    0    0    0
##     No covariate in the model.
##     Initial values
##             alp1 alp2 alp3
## Pop.CondInit    3    1    1
## ----------------------------------
## ----    Key algorithm options  ----
## ----------------------------------
##     Estimation of individual parameters (MAP)
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=300, K2=100
##     Number of chains:  1
##     Seed:  632545
##     Number of MCMC iterations for IS:  5000
##     Simulations:
##         nb of simulated datasets used for npde:  1000
##         nb of simulated datasets used for VPC:  100
##     Input/output
##         save the results to a file:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate
## [1,] alp1      2.28
## [2,] alp2      0.73
## [3,] alp3      0.76
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##      Parameter   Estimate
## alp1 omega2.alp1 0.34
## alp2 omega2.alp2 0.51
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##             omega2.alp1 omega2.alp2
## omega2.alp1 1           0
## omega2.alp2 0           1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
##
## Likelihood computed by importance sampling
##       -2LL= 3568.555
##       AIC = 3580.555
##       BIC = 3610.001
## ----------------------------------------------------
```

```r
ord.fit<-saemix.fit
plot(ord.fit, plot.type="convergence")
```



```
nsim<-1000
ordfit2<- simulateDiscreteSaemix(ord.fit, nsim=nsim)
discreteVPC(ordfit2, outcome="categorical")
```



Ordinal example with 7 categories (Lucie)

Count model

Example simulated by Lucie

- reasonable parameter estimates
  - launched several scenarios with 200 simulations each

```
# Count data model
countData.model<-function(psi,id,xidep) {
  tim <- xidep[,1]
  y <- xidep[,2]
  alpha <- psi[id,1]
  beta <- psi[id,2]
  lambda <- alpha*exp(-beta*tim)

  logpdf <- rep(0,length(tim))
  logpdf <- -lambda + y*( (log(alpha) - beta*tim )) - log(factorial(y))
  return(logpdf)
}

# Simulation function
countsimulate.poisson<-function(psi, id, xidep) {
```

```r
  tim <- xidep[,1]
  y <- xidep[,2]
  ymax<-max(y)
  alpha <- psi[id,1]
  beta <- psi[id,2]
  lambda <- alpha*exp(-beta*tim)
  y<-rpois(length(tim), lambda=lambda)
  y[y>ymax]<-ymax+1 # truncate to maximum observed value to avoid simulating aberrant values
  return(y)
}

# Settings
param <- c(39.1, 0.0388, 0.01 )
omega<-c(0.25, 0.25) # SD
paramSimul<-c(param, omega)
parnam<-c("alpha","beta","risk","omega.alpha","omega.beta")

nsuj<-40
xtim<-c(0.0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100)

partab<-as.data.frame(matrix(data=0,nrow=nsuj,ncol=2,dimnames=list(NULL,parnam[1:2])))
for(i in 1:2) partab[,i]<-rnorm(nsuj,mean=log(param[i]),sd=omega[i])
partab[(1+nsuj/2):nsuj,2]<-partab[(1+nsuj/2):nsuj,2]+param[3]
for(i in 1:2) partab[,i]<-exp(partab[,i])

psim<-data.frame()
for(itim in xtim) {
  lambda<-partab[,1]*exp(-partab[,2]*itim)
  psim<-rbind(psim,lambda)
}
datsim<-data.frame(id=rep(1:nsuj,each=length(xtim)),time=rep(xtim,nsuj),lambda=unlist(psim))
rownames(datsim)<-NULL
ysim<-rpois(dim(datsim)[1], lambda=datsim$lambda)
summary(datsim)
```

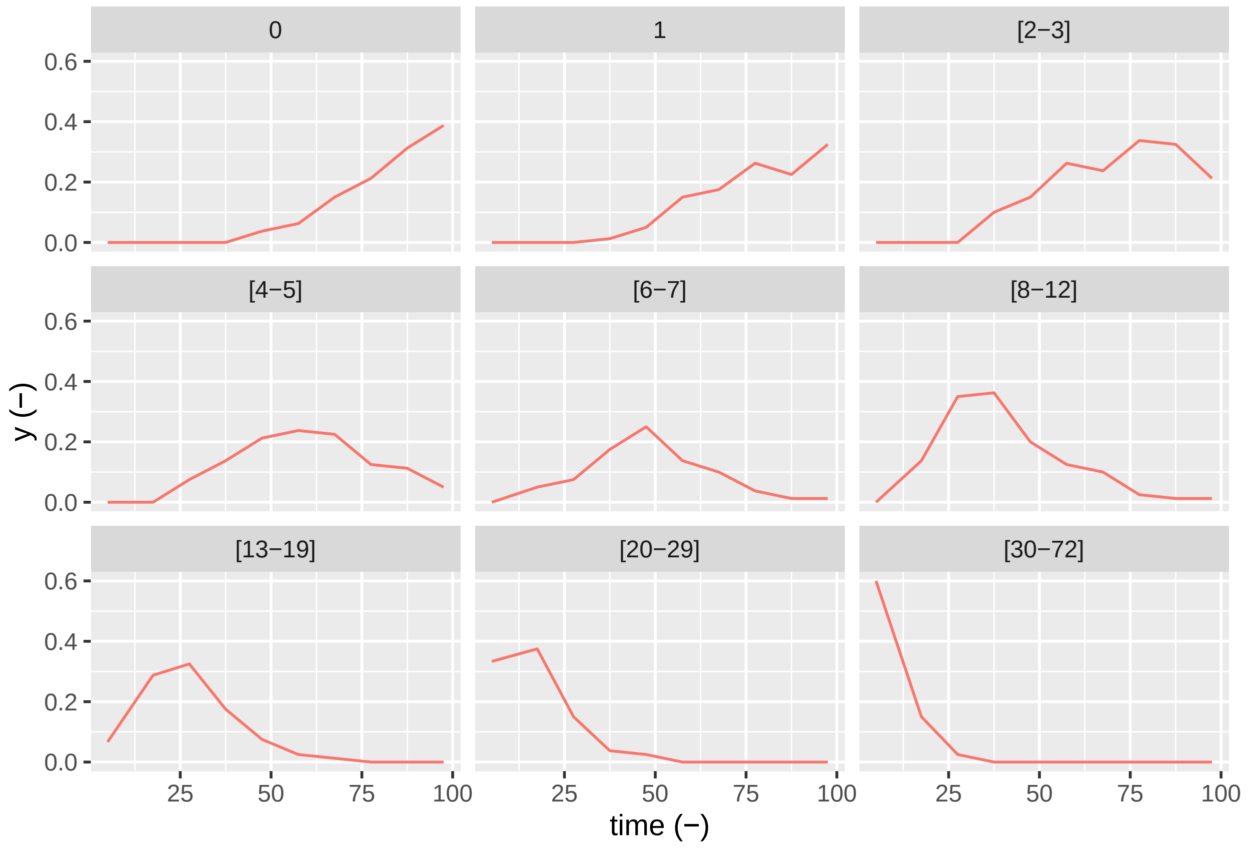
```
##        id             time         lambda
##  Min.   : 1.00   Min.   :  0   Min.   : 0.06446
##  1st Qu.:10.75   1st Qu.: 25   1st Qu.: 2.11011
##  Median :20.50   Median : 50   Median : 5.93171
##  Mean   :20.50   Mean   : 50   Mean   :10.92931
##  3rd Qu.:30.25   3rd Qu.: 75   3rd Qu.:16.28638
##  Max.   :40.00   Max.   :100   Max.   :65.67525
```

```r
datsim$y<-ysim
datsim$risk<-ifelse(datsim$id>(nsuj/2),1,0)

# Running saemix
saemix.data<-saemixData(name.data=datsim,
      name.group=c("id"),name.predictors=c("time","y"), name.covariates=c("risk"),name.X=c("time"))
```

```
## Column name(s)  do(es) not exist in the dataset, please check
## Remove columns 1 (  )
## No valid name given, attempting automatic recognition
## Automatic recognition of columns y successful
```

```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset datsim
##     Structured data: y ~ time + y | id
##     X variable for graphs: time ()
##     covariates: risk (-)
##       reference class for covariate risk :  0
```

```
plotDiscreteData(saemix.data, outcome="count")
```



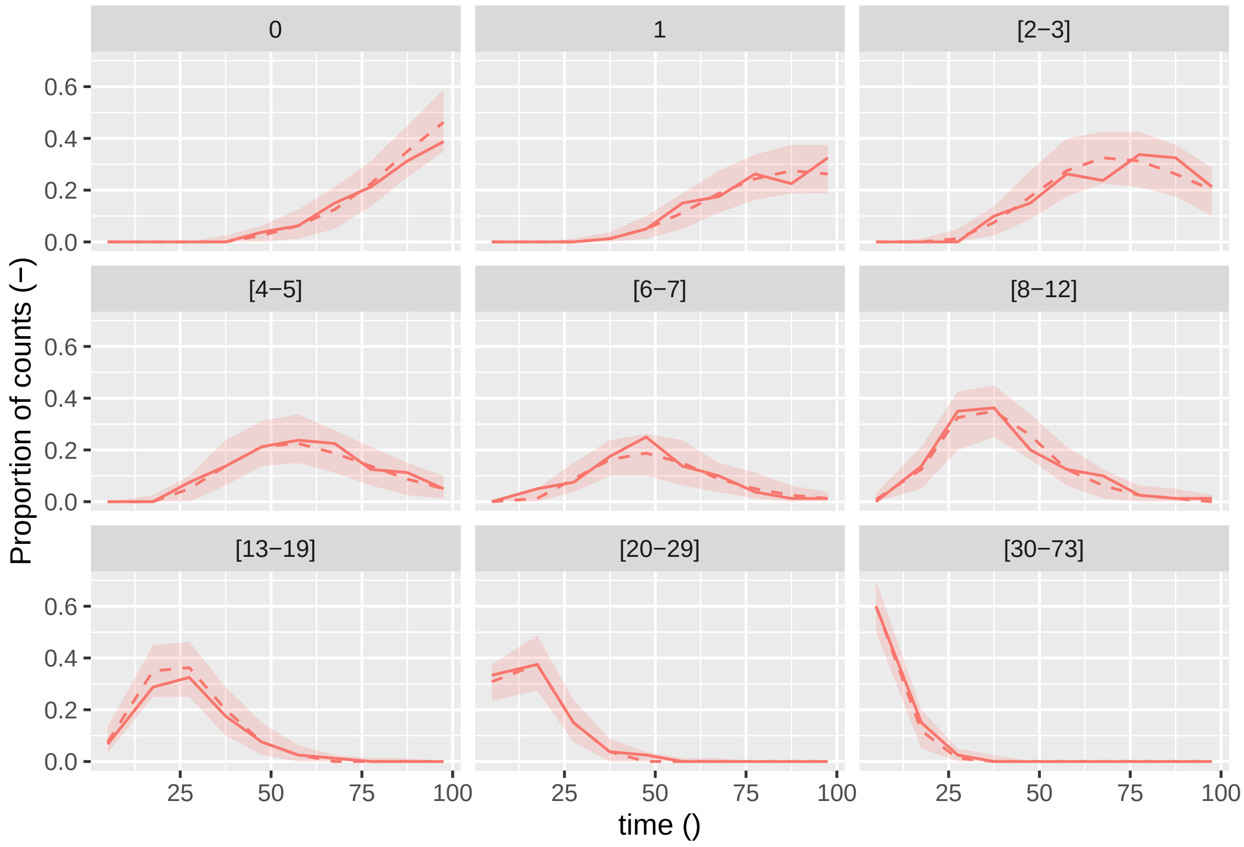
```
# Model
model.CountData<-saemixModel(model=countData.model,description="Count data model", modeltype="likelihoo
                             simulate.function = countsimulate.poisson,
                          psi0=matrix(c(param[1:2],0,param[3]),ncol=2,byrow=TRUE,dimnames=list(NULL,par
                          covariate.model=matrix(c(0,1),ncol=2), omega.init = diag(c(0.5,0.5)),
                          transform.par=c(1,1),covariance.model=matrix(c(1,0,0,1),ncol=2))
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  Count data model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   tim <- xidep[,1]
##   y <- xidep[,2]
##   alpha <- psi[id,1]
##   beta <- psi[id,2]
##   lambda <- alpha*exp(-beta*tim)
##
##   logpdf <- rep(0,length(tim))
##   logpdf <- -lambda + y*( (log(alpha) - beta*tim )) - log(factorial(y))
##   return(logpdf)
## }
##   Nb of parameters: 2
##       parameter names:  alpha beta
##       distribution:
##      Parameter Distribution Estimated
## [1,] alpha     log-normal   Estimated
## [2,] beta      log-normal   Estimated
##   Variance-covariance matrix:
##       alpha beta
## alpha     1    0
## beta      0    1
##   Covariate model:
##      alpha beta
## [1,]     0    1
##     Initial values
##              alpha   beta
## Pop.CondInit  39.1 0.0388
## Cov.CondInit   0.0 0.0100
```

```
#                          omega.init = diag(c(paramSimul[4]*3, paramSimul[5]**3)),
#                          transform.par=c(0,0),covariance.model=matrix(c(1,0,0,1),ncol=2))
# options
saemix.options<-list(seed=1234567, fim=FALSE, save=FALSE,save.graphs=FALSE, displayProgress=FALSE)
count.fit<-saemix(model.CountData,saemix.data,saemix.options)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data              ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset datsim
##     Structured data: y ~ time + y | id
##     X variable for graphs: time ()
##     covariates: risk (-)
##       reference class for covariate risk :  0
## Dataset characteristics:
##     number of subjects:     40
##     number of observations: 840
##     average/min/max nb obs: 21.00  /  21  /  21
## First 10 lines of data:
##    id time  y y.1 risk mdv cens occ ytype
## 1   1    0 30  30    0   0    0   1     1
## 2   1    5 25  25    0   0    0   1     1
## 3   1   10 21  21    0   0    0   1     1
## 4   1   15 17  17    0  0    0  1     1
## 5   1   20 18  18    0  0    0  1     1
## 6   1   25 16  16    0  0    0  1     1
## 7   1   30  8   8    0  0    0  1     1
## 8   1   35  5   5    0  0    0  1     1
## 9   1   40  9   9    0  0    0  1     1
## 10  1   45  6   6    0  0    0  1     1
## ----------------------------------
## ----          Model           ----
## ----------------------------------
## Nonlinear mixed-effects model
##   Model function:  Count data model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   tim <- xidep[,1]
##   y <- xidep[,2]
##   alpha <- psi[id,1]
##   beta <- psi[id,2]
##   lambda <- alpha*exp(-beta*tim)
##
##   logpdf <- rep(0,length(tim))
##   logpdf <- -lambda + y*( (log(alpha) - beta*tim )) - log(factorial(y))
##   return(logpdf)
## }
## <bytecode: 0x55eb232f03f0>
##   Nb of parameters: 2
##       parameter names:  alpha beta
##       distribution:
##      Parameter Distribution Estimated
## [1,] alpha     log-normal   Estimated
## [2,] beta      log-normal   Estimated
##   Variance-covariance matrix:
##       alpha beta
## alpha     1    0
## beta      0    1
##   Covariate model:
##      [,1] [,2]
## risk    0    1
##     Initial values
##              alpha   beta
## Pop.CondInit  39.1 0.0388
## Cov.CondInit   0.0 0.0100
## ----------------------------------
## ----    Key algorithm options  ----
## ----------------------------------
##     Estimation of individual parameters (MAP)
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=300, K2=100
##     Number of chains:  2
##     Seed:  1234567
##     Number of MCMC iterations for IS:  5000
##     Simulations:
##         nb of simulated datasets used for npde:  1000
##         nb of simulated datasets used for VPC:  100
##     Input/output
##         save the results to a file:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter       Estimate
## [1,] alpha           39.629
## [2,] beta             0.042
## [3,] beta_risk(beta) -0.109
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##       Parameter    Estimate
## alpha omega2.alpha 0.057
## beta  omega2.beta  0.051
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##              omega2.alpha omega2.beta
## omega2.alpha 1            0
## omega2.beta  0            1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
##
## Likelihood computed by importance sampling
##       -2LL= 3827.372
##       AIC = 3839.372
##       BIC = 3849.505
## ----------------------------------------------------
```

```
nsim<-200
count.fit<- simulateDiscreteSaemix(count.fit, nsim=nsim)
discreteVPC(count.fit, outcome="count")
```



TTE model

Assuming a Weibull model with shape parameter β and scale parameter λ, the hazard function is:

$$h(t) = \frac{\beta}{\lambda}\left(\frac{t}{\lambda}\right)^{\beta-1}$$

```
set.seed(12345)
xtim<-seq(0)
nsuj<-50
tte.data<-data.frame(id=rep(1:nsuj,each=length(xtim)),time=rep(xtim,nsuj))
psiM<-data.frame(lambda=seq(1.6,2,length.out=length(unique(tte.data$id))),beta = 2)

# TTE model
simul.tte<-function(psi,id,xidep) {
  T<-xidep
  N <- nrow(psi)
  Nj <- length(T)
  censoringtime = 3
  lambda <- psi[id,1]
  beta <- psi[id,2]
  obs <-rep(0,length(T))
  for (i in (1:N)){
    obs[id==i] <- rweibull(n=length(id[id==i]), shape=beta[i], scale=lambda[i])
  }
  obs[obs>censoringtime]<-censoringtime
  return(obs)
```

```r
}

preds <- simul.tte(psiM, tte.data$id, tte.data[,c("time")])
tte.data$y<-preds

tte.data$y<-0
tte.data$tlat<-preds
dat1<-tte.data[,c("id","time","y")]
dat2<-tte.data[,c("id","tlat","y")]
dat2$y<-as.integer(dat2$tlat>0 & dat2$tlat<3)
colnames(dat2)[2]<-"time"
tte.data<-rbind(dat1,dat2)
tte.data<-tte.data[order(tte.data$id, tte.data$time),]
head(tte.data)
```

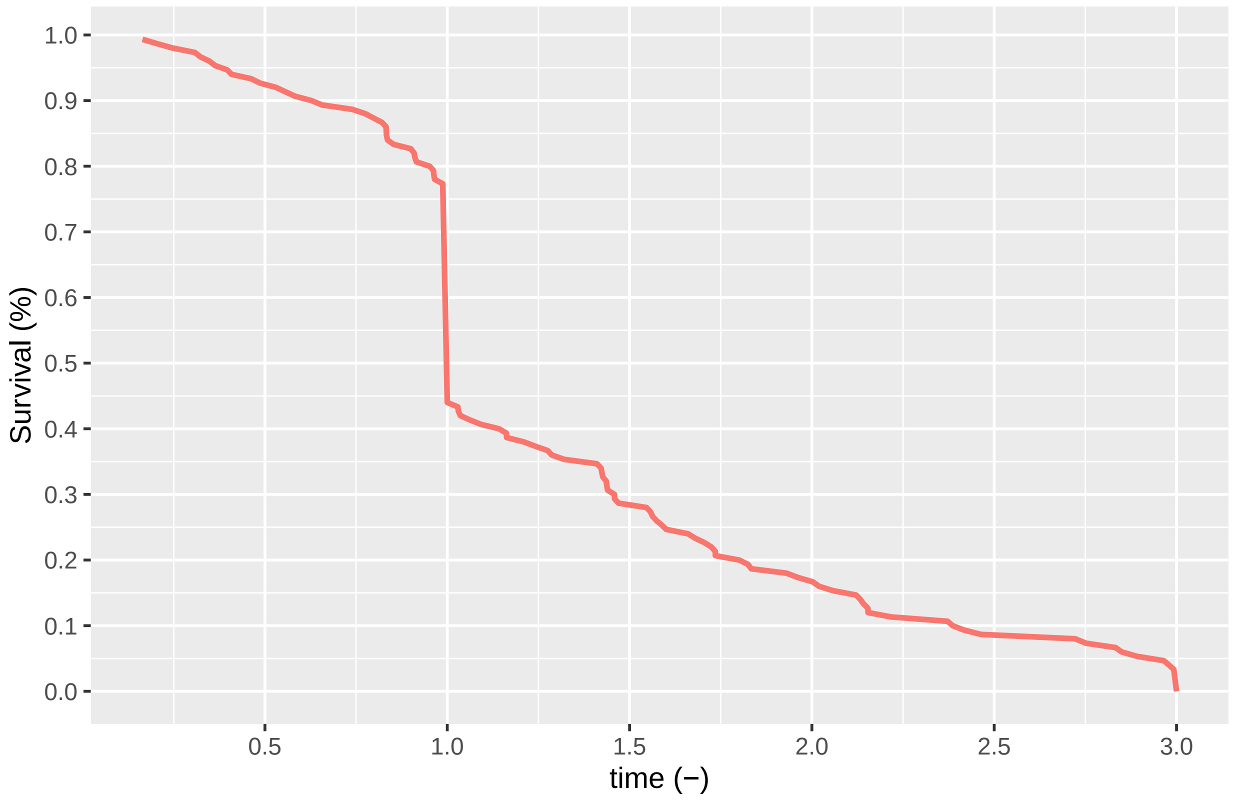
```
##     id      time y
## 2    1 0.0000000 0
## 102  1 0.5827343 1
## 101  1 0.9152915 1
## 1    1 1.0000000 0
## 4    2 0.0000000 0
## 104  2 0.5563256 1
```

```r
# Checking methods to simulate TTE data
if(FALSE) {
# Inverse probability method
  lambda<-1.5
  beta<-2
  nsim<-5000
  q1<-runif(nsim)
  tevent<-lambda*exp(log(-log(q1))/beta)
  tevent<-sort(tevent)
  summary(sort(tevent))
  plot(tevent, exp(-(tevent/lambda)^beta))
  # Using rweibull
  tevent2<-sort(rweibull(nsim, shape=beta, scale=lambda))

  plot(tevent, tevent2)
  abline(0,1)
  qqplot(tevent,tevent2)

  # Library survival to check fits
  library(survival)
  head(tevent2)
  summary(tevent2)
  mydat<-data.frame(tev=tevent2, status=rep(1,length(tevent2)))
  f1 <- survfit(Surv(tev, status) ~ 1, data = mydat)
  f1
  plot(f1)

  # From Stack exchange
  simulWeib <- function(N, lambda, rho, beta, rateC) {
    # covariate --> N Bernoulli trials
    x <- sample(x=c(0, 1), size=N, replace=TRUE, prob=c(0.5, 0.5))
    # Weibull latent event times
    v <- runif(n=N)
    Tlat <- (- log(v) / (lambda * exp(x * beta)))^(1 / rho)
    # censoring times
    C <- rexp(n=N, rate=rateC)
    # follow-up times and event indicators
    time <- pmin(Tlat, C)
    status <- as.numeric(Tlat <= C)
    # data set
    data.frame(id=1:N, tlat=Tlat, time=time, status=status, x=x)
  }
  dat1<-simulWeib(nsim, lambda=(1/lambda^beta), beta=1, rho=beta, rateC=1)
  summary(dat1$tlat)
  tevent3<-dat1$tlat
  tevent3<-sort(dat1$tlat)

  # Comparing all 3 survival functions with KM estimates => ok
  plot(f1)
  lines(tevent2, exp(-(tevent2/lambda)^beta), col="red",lwd=2)
  lines(tevent, exp(-(tevent/lambda)^beta), col="blue",lwd=2)
  lines(tevent3, exp(-(tevent3/lambda)^beta), col="green",lwd=2,lty=2)

}
```

```
saemix.data<-saemixData(name.data=tte.data, name.group=c("id"),
                        name.predictors=c("time"), name.response="y")
```

```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset tte.data
##     Structured data: y ~ time | id
##     Predictor: time ()
```

```
plotDiscreteData(saemix.data, outcome="tte")
```



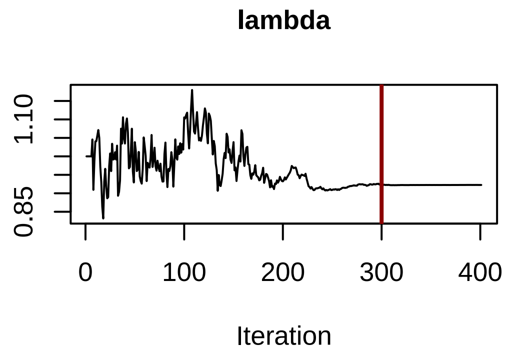
```
tte.model<-function(psi,id,xidep) {
  T<-xidep[,1]
  N <- nrow(psi)
  Nj <- length(T)
  # censoringtime = 6
  censoringtime = max(T)
  lambda <- psi[id,1]
  beta <- psi[id,2]
  init <- which(T==0)
  cens <- which(T==censoringtime)
  ind <- setdiff(1:Nj, append(init,cens))
  hazard <- (beta/lambda)*(T/lambda)^(beta-1)
  H <- (T/lambda)^beta
  logpdf <- rep(0,Nj)
  logpdf[cens] <- -H[cens] + H[cens-1]
  logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind])
  return(logpdf)
}

saemix.model<-saemixModel(model=tte.model,description="time model",modeltype="likelihood",
                          psi0=matrix(c(1,2),ncol=2,byrow=TRUE,dimnames=list(NULL,  c("lambda","beta")))
                          transform.par=c(1,1),covariance.model=matrix(c(1,0,0,0),ncol=2, byrow=TRUE))
```

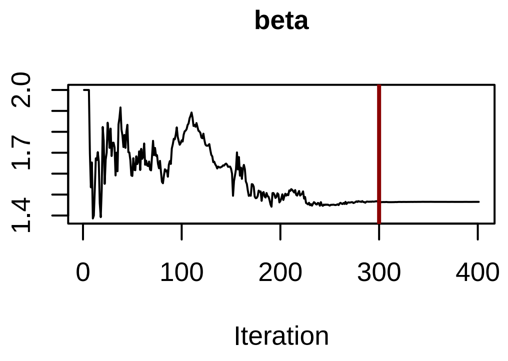
```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  time model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   T<-xidep[,1]
##   N <- nrow(psi)
##   Nj <- length(T)
##   # censoringtime = 6
##   censoringtime = max(T)
##   lambda <- psi[id,1]
##   beta <- psi[id,2]
##   init <- which(T==0)
##   cens <- which(T==censoringtime)
##   ind <- setdiff(1:Nj, append(init,cens))
##   hazard <- (beta/lambda)*(T/lambda)^(beta-1)
##   H <- (T/lambda)^beta
##   logpdf <- rep(0,Nj)
##   logpdf[cens] <- -H[cens] + H[cens-1]
##   logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind])
##   return(logpdf)
## }
##   Nb of parameters: 2
##       parameter names:  lambda beta
##       distribution:
##      Parameter Distribution Estimated
## [1,] lambda    log-normal   Estimated
## [2,] beta      log-normal   Estimated
##   Variance-covariance matrix:
##        lambda beta
## lambda      1    0
## beta        0    0
##     No covariate in the model.
##     Initial values
##              lambda beta
## Pop.CondInit      1    2
```

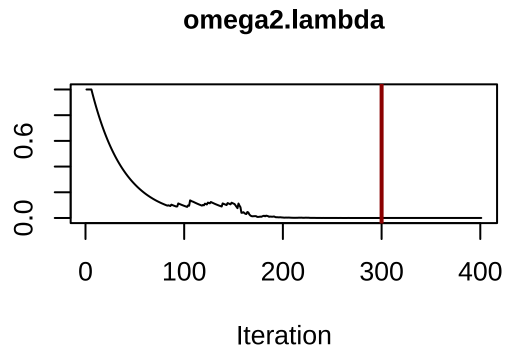
```
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE)
tte.fit<-saemix(saemix.model,saemix.data,saemix.options)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data             ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset tte.data
##     Structured data: y ~ time | id
##     Predictor: time ()
## Dataset characteristics:
##     number of subjects:     50
##     number of observations: 200
##     average/min/max nb obs: 4.00  /  4  /  4
## First 10 lines of data:
##     id      time y mdv cens occ ytype
## 2    1 0.0000000 0   0    0   1     1
## 102  1 0.5827343 1   0    0   1     1
## 101  1 0.9152915 1   0    0   1     1
## 1    1 1.0000000 0   0    0   1     1
## 4    2 0.0000000 0   0    0   1     1
## 104  2 0.5563256 1   0    0   1     1
## 103  2 0.8362126 1   0    0   1     1
## 3    2 1.0000000 0   0    0   1     1
## 6    3 0.0000000 0   0    0   1     1
## 5    3 1.0000000 0   0    0   1     1
## ----------------------------------
## ----          Model           ----
## ----------------------------------
## Nonlinear mixed-effects model
##   Model function:  time model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   T<-xidep[,1]
##   N <- nrow(psi)
##   Nj <- length(T)
##   # censoringtime = 6
##   censoringtime = max(T)
##   lambda <- psi[id,1]
##   beta <- psi[id,2]
##   init <- which(T==0)
##   cens <- which(T==censoringtime)
##   ind <- setdiff(1:Nj, append(init,cens))
##   hazard <- (beta/lambda)*(T/lambda)^(beta-1)
##   H <- (T/lambda)^beta
##   logpdf <- rep(0,Nj)
##   logpdf[cens] <- -H[cens] + H[cens-1]
##   logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind])
##   return(logpdf)
## }
## <bytecode: 0x55eb1d5e52a8>
##   Nb of parameters: 2
##       parameter names:  lambda beta
##       distribution:
##      Parameter Distribution Estimated
## [1,] lambda    log-normal   Estimated
## [2,] beta      log-normal   Estimated
##   Variance-covariance matrix:
##        lambda beta
## lambda      1    0
## beta        0    0
##     No covariate in the model.
##     Initial values
##              lambda beta
## Pop.CondInit      1    2
## ----------------------------------
## ----    Key algorithm options  ----
## ----------------------------------
##     Estimation of individual parameters (MAP)
##     Estimation of standard errors and linearised log-likelihood
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=300, K2=100
##     Number of chains:  1
##     Seed:  632545
##     Number of MCMC iterations for IS:  5000
##     Simulations:
##         nb of simulated datasets used for npde:  1000
##         nb of simulated datasets used for VPC:  100
##     Input/output
##         save the results to a file:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate SE   CV(%)
## [1,] lambda    0.92     0.19 21
## [2,] beta      1.47     0.29 19
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##        Parameter     Estimate SE   CV(%)
## lambda omega2.lambda 0.00011  0.22 2e+05
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##               omega2.lambda
## omega2.lambda 1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= 620.2125
##       AIC = 628.2125
##       BIC = 635.8606
##
## Likelihood computed by importance sampling
##       -2LL= 140.5422
##       AIC = 148.5422
##       BIC = 156.1903
## ----------------------------------------------------
```

```r
plot(tte.fit, plot.type="convergence")
```







RTTE model

For repeated time-to-events, a simulation algorithm is to simulate TTE repeatedly in an individual until the time to censoring has been reached (Penichou et al. 2014). Here the simulation for each successive event is perfomed by simulating a random variable in $\mathcal{U}[0, 1]$ and using the inverse of the cumulative density function to generate the corresponding time to event.

```
# simulate latent times for RTTE data - ne marche pas du tout
simul.rtte.rweib<-function(psi,id,xidep) {
  T<-xidep
  N <- nrow(psi)
  Nj <- length(T)
  censoringtime = 3
  lambda <- psi[,1]
  beta <- psi[,2]
  obs <-rep(0,length(T))
  for (i in (1:N)){
    obs[id==i] <- cumsum(rweibull(n=length(id[id==i]), shape=beta[i], scale=lambda[i])) # simulate delt
    if(max(obs[id==i])<censoringtime) message("Increase the number of visits\n")
  }
  obs[obs>censoringtime]<-censoringtime
  return(obs)
}


simul.rtte.unif<-function(psi) { # xidep, id not important, we only use psi
  censoringtime <- 3
  maxevents <- 30
  lambda <- psi[,1]
```

```r
  beta <- psi[,2]
  simdat<-NULL
  N<-nrow(psi)
  for(i in 1:N) {
    eventTimes<-c(0)
    T<-0
    Vj<-runif(1)
#     T <- (-log(Vj)*lambda[i])^(beta[i])
      T<-lambda[i]*(-log(Vj))^(1/beta[i])
    nev<-0
    while (T < censoringtime & nev<maxevents){
      eventTimes <- c(eventTimes, T)
      nev<-nev+1
      Vj<-runif(1)
#       T <- T+(-log(Vj)*lambda[i])^(beta[i])
#       T<-(-log(Vj)*lambda[i] + T^(1/beta[i]))^(beta[i])
      T<-lambda[i]*(-log(Vj) + (T/lambda[i])^(beta[i]))^(1/beta[i])
    }
    if(nev==maxevents) {
      message("Reached maximum number of events\n")
    }
    eventTimes<-c(eventTimes, censoringtime)
    cens<-rep(1,length(eventTimes))
    cens[1]<-cens[length(cens)]<-0
    simdat<-rbind(simdat,
                  data.frame(id=i, T=eventTimes, status=cens))
  }
  return(simdat)
}

# Subjects
set.seed(12345)
param<-c(2, 1.5, 0.5)
# param<-c(4, 1.2, 0.3)
omega<-c(0.25,0.25)
nsuj<-200
risk<-rep(0,nsuj)
risk[(nsuj/2+1):nsuj]<-1
psiM<-data.frame(lambda=param[1]*exp(rnorm(nsuj,sd=omega[1])), beta=param[2]*exp(param[3]*risk+rnorm(nsu
simdat <- simul.rtte.unif(psiM)
```

```
## Reached maximum number of events
```

```r
simdat$risk<-as.integer(simdat$id>(nsuj/2))


if(FALSE) { # Check simulated parameters
  summary(psiM)
  apply(log(psiM),2,sd)
}

if(FALSE) {

xtim<-seq(0:20)
rtte.data<-data.frame(id=rep(1:nsuj,each=length(xtim)),time=rep(xtim,nsuj))
preds <- simul.rtte.rweib(psiM, rtte.data$id, rtte.data[,c("time")])

par(mfrow=c(1,2))
hist(preds)
hist(simdat$T[simdat$T>0])

  rtte.data$tlat<-preds
  rtte.data$status<-as.integer(rtte.data$tlat<3)

  # Remove duplicated censored times
  dat1<-NULL
  for(i in 1:nsuj) {
    idat<-rtte.data[rtte.data$id==i,]
    idat<-idat[!duplicated(idat$tlat),,drop=FALSE]
    dat1<-rbind(dat1, c(i,0,0), idat[,-c(2)])
  }
  rtte.data<-dat1
  table(tapply(rtte.data$id, rtte.data$id, length))
}

if(FALSE)
  write.table(simdat,file.path(ecoDir,"simulatedRTTE.csv"), quote=F, row.names=F)
```

- Corrected simulation file thanks to Lucie ⇒ now getting better estimates for the parameters (at least for N=200)
  - still completely different from using rweibull so really need to use the inverse cdf method to simulate
- set up simulation files for different scenarios to check out performances
  - ‘many’ events (3-4/subject)
  - ‘few’ events (1-2/subject)
  - N=200: few with IIV=25%, many with IIV=25 and 50%
  - N=50: many with IIV=25%
  - also a simulation mimicking Lucie’s (censoring time=50, risk on lambda)

```r
saemix.data<-saemixData(name.data=simdat, name.group=c("id"), name.predictors=c("T"), name.response="sta
```

```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset simdat
##     Structured data: status ~ T | id
##     Predictor: T ()
##     covariates: risk (-)
##       reference class for covariate risk :  0
```

```r
rtte.model<-function(psi,id,xidep) {
  T<-xidep[,1]
  N <- nrow(psi) # nb of subjects
  Nj <- length(T) # nb of events (including 0 and censoring times)
  # censoringtime = 6
  censoringtime = max(T) # same censoring for everyone
lambda <- psi[id,1]
beta <- psi[id,2]
tinit <- which(T==0) # indices of beginning of observation period
tcens <- which(T==censoringtime) # indices of censored events
tevent <- setdiff(1:Nj, append(tinit,tcens)) # indices of non-censored event times
hazard <- (beta/lambda)*(T/lambda)^(beta-1)
H <- (T/lambda)^beta
logpdf <- rep(0,Nj)
logpdf[tcens] <- -H[tcens] + H[tcens-1]
logpdf[tevent] <- -H[tevent] + H[tevent-1] + log(hazard[tevent])
return(logpdf)
}

saemix.model.base<-saemixModel(model=rtte.model,description="Repeated TTE model",modeltype="likelihood"
                        psi0=matrix(c(1,2),ncol=2,byrow=TRUE,dimnames=list(NULL,  c("lambda","beta"))
                        transform.par=c(1,1),covariance.model=matrix(c(1,0,0,1),ncol=2, byrow=TRUE))
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  Repeated TTE model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   T<-xidep[,1]
##   N <- nrow(psi) # nb of subjects
##   Nj <- length(T) # nb of events (including 0 and censoring times)
##   # censoringtime = 6
##   censoringtime = max(T) # same censoring for everyone
##   lambda <- psi[id,1]
##   beta <- psi[id,2]
##   tinit <- which(T==0) # indices of beginning of observation period
##   tcens <- which(T==censoringtime) # indices of censored events
##   tevent <- setdiff(1:Nj, append(tinit,tcens)) # indices of non-censored event times
##   hazard <- (beta/lambda)*(T/lambda)^(beta-1)
##   H <- (T/lambda)^beta
##   logpdf <- rep(0,Nj)
##   logpdf[tcens] <- -H[tcens] + H[tcens-1]
##   logpdf[tevent] <- -H[tevent] + H[tevent-1] + log(hazard[tevent])
##   return(logpdf)
## }
##   Nb of parameters: 2
##       parameter names:  lambda beta
##       distribution:
##      Parameter Distribution Estimated
## [1,] lambda    log-normal   Estimated
## [2,] beta      log-normal   Estimated
##   Variance-covariance matrix:
##        lambda beta
## lambda      1    0
## beta        0    1
##     No covariate in the model.
##     Initial values
##               lambda beta
## Pop.CondInit      1    2
```

```
saemix.model<-saemixModel(model=rtte.model,description="Repeated TTE model",modeltype="likelihood",
                          psi0=matrix(c(1,2),ncol=2,byrow=TRUE,dimnames=list(NULL,  c("lambda","beta"))),
                          transform.par=c(1,1),covariate.model=matrix(c(0,1),ncol=2),
                          covariance.model=matrix(c(1,0,0,1),ncol=2, byrow=TRUE))
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  Repeated TTE model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   T<-xidep[,1]
##   N <- nrow(psi) # nb of subjects
##   Nj <- length(T) # nb of events (including 0 and censoring times)
##   # censoringtime = 6
##   censoringtime = max(T) # same censoring for everyone
##   lambda <- psi[id,1]
##   beta <- psi[id,2]
##   tinit <- which(T==0) # indices of beginning of observation period
##   tcens <- which(T==censoringtime) # indices of censored events
##   tevent <- setdiff(1:Nj, append(tinit,tcens)) # indices of non-censored event times
##   hazard <- (beta/lambda)*(T/lambda)^(beta-1)
##   H <- (T/lambda)^beta
##   logpdf <- rep(0,Nj)
##   logpdf[tcens] <- -H[tcens] + H[tcens-1]
##   logpdf[tevent] <- -H[tevent] + H[tevent-1] + log(hazard[tevent])
##   return(logpdf)
## }
##   Nb of parameters: 2
##       parameter names:  lambda beta
##       distribution:
##      Parameter Distribution Estimated
## [1,] lambda    log-normal   Estimated
## [2,] beta      log-normal   Estimated
##   Variance-covariance matrix:
##        lambda beta
## lambda      1    0
## beta        0    1
##   Covariate model:
##      lambda beta
## [1,]      0    1
##     Initial values
##               lambda beta
## Pop.CondInit      1    2
## Cov.CondInit      0    0
```

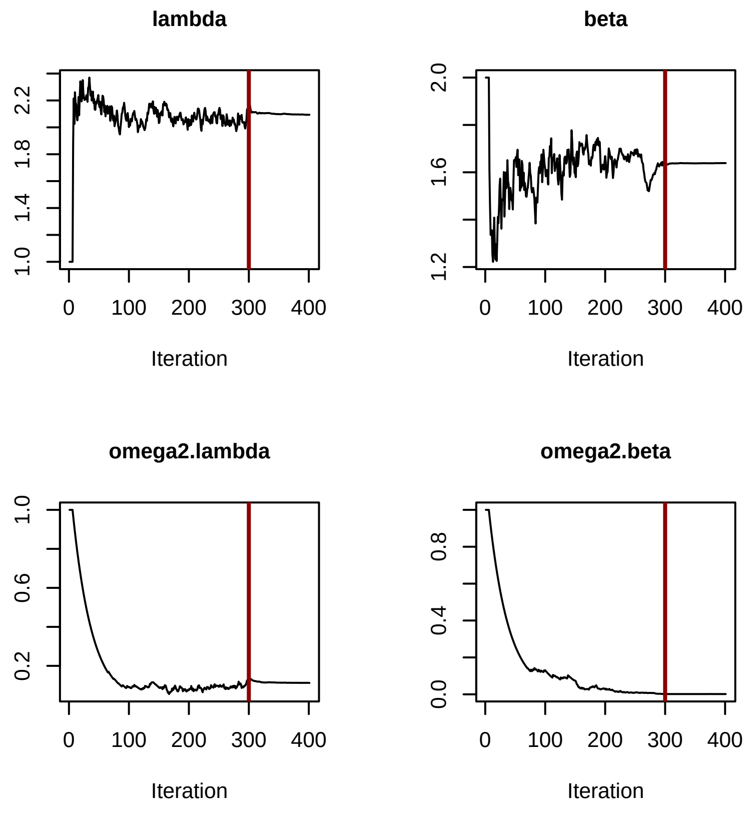
```
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, fim=FALSE, displayProgress=FALSE)
rtte.fit<-saemix(saemix.model,saemix.data,saemix.options)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## ----------------------------------
## ----          Data            ----
## ----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset simdat
##     Structured data: status ~ T | id
##     Predictor: T ()
##     covariates: risk (-)
##       reference class for covariate risk :  0
## Dataset characteristics:
##     number of subjects:     200
##     number of observations: 967
##     average/min/max nb obs: 4.83  /  2  /  32
## First 10 lines of data:
##    id         T status risk mdv cens occ ytype
## 1   1 0.0000000      0    0   0    0   1     1
## 2   1 0.7520145      1    0   0    0   1     1
## 3   1 0.8775847      1    0   0    0   1     1
## 4   1 2.4331650      1    0   0    0   1     1
## 5   1 3.0000000      0    0   0    0   1     1
## 6   2 0.0000000      0    0   0    0   1     1
## 7   2 1.3712351      1    0   0    0   1     1
## 8   2 3.0000000      0    0   0    0   1     1
## 9   3 0.0000000      0    0   0    0   1     1
## 10  3 2.8564910      1    0   0    0   1     1
## ----------------------------------
## ----          Model           ----
## ----------------------------------
## Nonlinear mixed-effects model
##   Model function:  Repeated TTE model
##   Model type:  likelihood
## function(psi,id,xidep) {
##   T<-xidep[,1]
##   N <- nrow(psi) # nb of subjects
##   Nj <- length(T) # nb of events (including 0 and censoring times)
##   # censoringtime = 6
##   censoringtime = max(T) # same censoring for everyone
##   lambda <- psi[id,1]
##   beta <- psi[id,2]
##   tinit <- which(T==0) # indices of beginning of observation period
##   tcens <- which(T==censoringtime) # indices of censored events
##   tevent <- setdiff(1:Nj, append(tinit,tcens)) # indices of non-censored event times
##   hazard <- (beta/lambda)*(T/lambda)^(beta-1)
##   H <- (T/lambda)^beta
##   logpdf <- rep(0,Nj)
##   logpdf[tcens] <- -H[tcens] + H[tcens-1]
##   logpdf[tevent] <- -H[tevent] + H[tevent-1] + log(hazard[tevent])
##   return(logpdf)
## }
## <bytecode: 0x55eb2327f380>
##   Nb of parameters: 2
##       parameter names:  lambda beta
##       distribution:
```

```
##      Parameter Distribution Estimated
## [1,] lambda    log-normal   Estimated
## [2,] beta      log-normal   Estimated
##   Variance-covariance matrix:
##        lambda beta
## lambda      1    0
## beta        0    1
##   Covariate model:
##      [,1] [,2]
## risk    0    1
##     Initial values
##             lambda beta
## Pop.CondInit      1    2
## Cov.CondInit      0    0
## ----------------------------------
## ----    Key algorithm options  ----
## ----------------------------------
##     Estimation of individual parameters (MAP)
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=300, K2=100
##     Number of chains:  1
##     Seed:  632545
##     Number of MCMC iterations for IS:  5000
##     Simulations:
##         nb of simulated datasets used for npde:  1000
##         nb of simulated datasets used for VPC:  100
##     Input/output
##         save the results to a file:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter       Estimate
## [1,] lambda          2.1
## [2,] beta            1.6
## [3,] beta_risk(beta) 0.4
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##        Parameter     Estimate
## lambda omega2.lambda 0.1125
## beta   omega2.beta   0.0015
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##               omega2.lambda omega2.beta
## omega2.lambda 1             0
## omega2.beta   0             1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
##
```

```
## Likelihood computed by importance sampling
##       -2LL= 690.2485
##       AIC = 702.2485
##       BIC = 722.0384
## ------------------------------------------------------
```

```
plot(rtte.fit, plot.type="convergence")
```



```
# Weibull model re-initialised at each event time => gives wacky results
rtte.model2<-function(psi,id,xidep) {
  T<-xidep[,1]
  N <- nrow(psi) # nb of subjects
  Nj <- length(T) # nb of events (including 0 and censoring times)
  # censoringtime = 6
  censoringtime = max(T) # same censoring for everyone
  lambda <- psi[id,1]
  beta <- psi[id,2]
  init <- which(T==0) # indices of beginning of observation period
  cens <- which(T==censoringtime) # indices of censored events
  ind <- setdiff(1:Nj, append(init,cens)) # indices of non-censored event times
  Tdiff<-c(0,T[2:Nj]-T[1:(Nj-1)])
  Tdiff[T==0]<-0
  hazard <- (beta/lambda)*(Tdiff/lambda)^(beta-1)
  H <- (Tdiff/lambda)^beta
  logpdf <- rep(0,Nj)
  logpdf[cens] <- -H[cens] + H[cens-1]
  logpdf[ind] <- -H[ind] + H[ind-1] + log(hazard[ind])
  return(logpdf)
}
```

Maud’s algorithm for covariate selection

- compare function
  - modified so it accepts either a list or several arguments (if several arguments, it tests which are models and compares those)
  - changed stop into a return to avoid the function failing and entering debug mode
- test functions in **testmaud** folder
  - functions computing BIC
  - function comparing models
  - function testing stepwise algorithm
- in the code below (uncomment FALSE to execute), res.forward and res.backward lead to the same model (Weight on ka and V)
- **TODO**: modify the optimisation step for fixed effects to avoid the warning message from optim()
- summary from stepwise algorithm not particularly easy to read
  - **TODO** modify ?
  - documentation: what does the algorithm do (in particular ‘both’) and how ?
- check why covariate Sex does not enter when using a forward selection ?
- getting a weird bug (saemixObject[“data”]: objet de type ‘S4’ non indiçable) => try relaunching from a clean session… OK

```
if(!testMode) {
  source(file.path(progDir,"backward.R"))
  source(file.path(progDir,"forward.R"))
  source(file.path(progDir,"stepwise.R"))
  source(file.path(progDir,"func_stepwise.R"))
  source(file.path(progDir,"func_compare.R"))
}
theo.saemix<-read.table(file.path(datDir, "theo.saemix.tab"), header=T)

saemix.data<-saemixData(name.data=theo.saemix,header=TRUE,sep=" ",na=NA,
  name.group=c("Id"),name.predictors=c("Dose","Time"),
  name.response=c("Concentration"),name.covariates=c("Weight","Sex"),
  units=list(x="hr",y="mg/L",covariates=c("kg","-")), name.X="Time")
```

```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset theo.saemix
##     Structured data: Concentration ~ Dose + Time | Id
##     X variable for graphs: Time (hr)
##     covariates: Weight (kg), Sex (-)
##       reference class for covariate Sex :  0
```

```
# Definition of models to be compared
model1cpt<-function(psi,id,xidep) {
   dose<-xidep[,1]
   tim<-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)
```

```
}

saemix.model1<-saemixModel(model=model1cpt,modeltype="structural",
                           description="One-compartment model with first-order absorption",
                           psi0=matrix(c(1.,20,0.5,0.1,0,-0.01),ncol=3,byrow=TRUE, dimnames=list(NULL, c
                           transform.par=c(1,1,1),covariate.model=matrix(c(0,0,1,0,0,0),ncol=3,byrow=TRUE
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption
##   Model type:  structural
## function(psi,id,xidep) {
##    dose<-xidep[,1]
##    tim<-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)
## }
##   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: constant , initial values: a.1=1
##   Covariate model:
##      ka V CL
## [1,]  0 0  1
## [2,]  0 0  0
##     Initial values
##               ka  V    CL
## Pop.CondInit 1.0 20  0.50
## Cov.CondInit 0.1  0 -0.01
```

```
saemix.model2<-saemixModel(model=model1cpt,modeltype="structural",
                           description="One-compartment model with first-order absorption",
                           psi0=matrix(c(1.,20,0.5,0.1,0,-0.01),ncol=3,byrow=TRUE, dimnames=list(NULL, c
                           transform.par=c(1,1,1),covariate.model=matrix(rep(1,6),ncol=3,byrow=TRUE))
```

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

```
##
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption
##   Model type:  structural
## function(psi,id,xidep) {
##    dose<-xidep[,1]
##    tim<-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)
## }
##   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: constant , initial values: a.1=1
##   Covariate model:
##      ka V CL
## [1,]  1 1  1
## [2,]  1 1  1
##     Initial values
##               ka  V    CL
## Pop.CondInit 1.0 20  0.50
## Cov.CondInit 0.1  0 -0.01
```

```
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE, warnings=FALSE)
if(FALSE) {
  saemix.fit1<-saemix(saemix.model1,saemix.data,saemix.options)
  saemix.fit2<-saemix(saemix.model2,saemix.data,saemix.options)
  covariate.init <- matrix(c(1,0,0,0,1,0),ncol=3,nrow=2)
  res.forward <- step.saemix(saemix.fit1, direction = "forward")
  res.backward <- step.saemix(saemix.fit2, direction = "backward")# , covariate.init=covariate.init)
  res.stepwise <- step.saemix(saemix.fit1, direction="both", covariate.init=covariate.init)
}
```

Running saemix with only one IIV or parameter

- only one IIV
  - works
- only one parameter
  - old workaround: use a dummy parameter fixed
  - Belhal fixed the issue

```r
# One IIV
saemix.model1iiv<-saemixModel(model=model1cpt,modeltype="structural",
                          description="One-compartment model with first-order absorption",
                          psi0=matrix(c(1.,20,0.5,0.1,0,-0.01),ncol=3,byrow=TRUE, dimnames=list(NULL, c
                          transform.par=c(1,1,1),covariance.model=matrix(c(0,0,0,0,0,0,0,0,1),ncol=3,by
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption
##   Model type:  structural
## function(psi,id,xidep) {
##    dose<-xidep[,1]
##    tim<-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)
## }
##   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  0 0  0
## V   0 0  0
## CL  0 0  1
##   Error model: constant , initial values: a.1=1
##     No covariate in the model.
##     Initial values
##               ka  V    CL
## Pop.CondInit 1.0 20  0.50
## Cov.CondInit 0.1  0 -0.01
```

```r
saemix.fit1iiv<-saemix(saemix.model1iiv,saemix.data,saemix.options)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data             ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset theo.saemix
##     Structured data: Concentration ~ Dose + Time | Id
##     X variable for graphs: Time (hr)
##     covariates: Weight (kg), Sex (-)
##       reference class for covariate Sex :  0
## Dataset characteristics:
##     number of subjects:     12
##     number of observations: 120
##     average/min/max nb obs: 10.00  /  10  /  10
## First 10 lines of data:
##    Id    Dose  Time Concentration Weight Sex mdv cens occ ytype
## 1   1 319.992  0.25          2.84   79.6   1   0    0   1     1
## 2   1 319.992  0.57          6.57   79.6   1   0    0   1     1
## 3   1 319.992  1.12         10.50   79.6   1   0    0   1     1
## 4   1 319.992  2.02          9.66   79.6   1   0    0   1     1
## 5   1 319.992  3.82          8.58   79.6   1   0    0   1     1
## 6   1 319.992  5.10          8.36   79.6   1   0    0   1     1
## 7   1 319.992  7.03          7.47   79.6   1   0    0   1     1
## 8   1 319.992  9.05          6.89   79.6   1   0    0   1     1
## 9   1 319.992 12.12          5.94   79.6   1   0    0   1     1
## 10  1 319.992 24.37          3.28   79.6   1   0    0   1     1
## -----------------------------------
## ----          Model            ----
## -----------------------------------
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption
##   Model type:  structural
## function(psi,id,xidep) {
##    dose<-xidep[,1]
##    tim<-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: 0x55eb1d39e888>
##   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  0 0  0
## V   0 0  0
## CL  0 0  1
##   Error model: constant , initial values: a.1=1
##     No covariate in the model.
##     Initial values
##              ka  V  CL
## Pop.CondInit  1 20 0.5
## -----------------------------------
## ----    Key algorithm options  ----
## -----------------------------------
##     Estimation of individual parameters (MAP)
##     Estimation of standard errors and linearised log-likelihood
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=300, K2=100
##     Number of chains:  5
##     Seed:  632545
##     Number of MCMC iterations for IS:  5000
##     Simulations:
##         nb of simulated datasets used for npde:  1000
##         nb of simulated datasets used for VPC:  100
##     Input/output
##         save the results to a file:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate SE    CV(%)
## [1,] ka         1.5     0.144  9.7
## [2,] V         31.8     1.292  4.1
## [3,] CL         2.7     0.309 11.4
## [4,] a.1        1.2     0.084  6.8
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate SE    CV(%)
## CL omega2.CL 0.11     0.057 52
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.CL
## omega2.CL 1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= 409.278
##       AIC = 419.278
##       BIC = 421.7025
##
## Likelihood computed by importance sampling
##       -2LL= 409.1383
##       AIC = 419.1383
##       BIC = 421.5629
## ----------------------------------------------------
```

```
# Fails - no IIV
saemix.modelNoIIV<-saemixModel(model=model1cpt,modeltype="structural",
                          description="One-compartment model with first-order absorption",
                          psi0=matrix(c(1.,20,0.5,0.1,0,-0.01),ncol=3,byrow=TRUE, dimnames=list(NULL, c
                          transform.par=c(1,1,1),covariance.model=matrix(c(0,0,0,0,0,0,0,0,0),ncol=3,by
```

```
## At least one parameter should have IIV in the model, the covariance model may not be only 0s.
## Error initialising SaemixModel object:

## [ SaemixModel : Error ]

## At least one parameter with IIV must be included in the model.

## Error in validObject(xmod) :
##   objet de classe "SaemixModel" incorrect: Invalid IIV structure
## [1] "Creation of SaemixModel failed"
```

```
# Fails - one parameter with IIV but this parameter is not estimated
saemix.modelNoIIV<-saemixModel(model=model1cpt,modeltype="structural",
                          description="One-compartment model with first-order absorption",
                          psi0=matrix(c(1.,20,0.5,0.1,0,-0.01),ncol=3,byrow=TRUE, dimnames=list(NULL, c
                          transform.par=c(1,1,1),covariance.model=matrix(c(0,0,0,0,0,0,0,0,1),ncol=3,by
```

```
## Error initialising SaemixModel object:

## [ SaemixModel : Error ]

## At least one parameter with IIV must be estimated and not fixed in the model.

## Error in validObject(xmod) :
##   objet de classe "SaemixModel" incorrect: Invalid IIV structure
## [1] "Creation of SaemixModel failed"
```

```
# One parameter only

model1cpt.onepar<-function(psi,id,xidep) {
   dose<-xidep[,1]
   tim<-xidep[,2]
   ka<-1.5
   V<-30
   CL<-psi[id,1]
   k<-CL/V
   ypred<-dose*ka/(V*(ka-k))*(exp(-k*tim)-exp(-ka*tim))
   return(ypred)
}
saemix.modelonepar<-saemixModel(model=model1cpt.onepar,modeltype="structural",
                          description="One-compartment model with first-order absorption, only CL",
                          psi0=matrix(c(0.5,0),ncol=1,byrow=TRUE, dimnames=list(NULL, c("CL"))),
                          transform.par=c(1))
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption, only CL
##   Model type:  structural
## function(psi,id,xidep) {
##    dose<-xidep[,1]
##    tim<-xidep[,2]
##    ka<-1.5
##    V<-30
##    CL<-psi[id,1]
##    k<-CL/V
##    ypred<-dose*ka/(V*(ka-k))*(exp(-k*tim)-exp(-ka*tim))
##    return(ypred)
## }
##   Nb of parameters: 1
##       parameter names:  CL
##       distribution:
##      Parameter Distribution Estimated
## [1,] CL        log-normal   Estimated
##   Variance-covariance matrix:
##    CL
## CL  1
##   Error model: constant , initial values: a.1=1
##     No covariate in the model.
##     Initial values
##               CL
## Pop.CondInit 0.5
## Cov.CondInit 0.0
```

```
saemix.onepar<-saemix(saemix.modelonepar,saemix.data,saemix.options)
```

```
## Nonlinear mixed-effects model fit by the SAEM algorithm
## -----------------------------------
## ----          Data              ----
## -----------------------------------
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset theo.saemix
##     Structured data: Concentration ~ Dose + Time | Id
##     X variable for graphs: Time (hr)
##     covariates: Weight (kg), Sex (-)
##       reference class for covariate Sex :  0
## Dataset characteristics:
##     number of subjects:     12
##     number of observations: 120
##     average/min/max nb obs: 10.00  /  10  /  10
## First 10 lines of data:
##    Id    Dose  Time Concentration Weight Sex mdv cens occ ytype
## 1   1 319.992  0.25          2.84   79.6   1   0    0   1     1
## 2   1 319.992  0.57          6.57   79.6   1   0    0   1     1
## 3   1 319.992  1.12         10.50   79.6   1   0    0   1     1
## 4   1 319.992  2.02          9.66   79.6   1   0    0   1     1
## 5   1 319.992  3.82          8.58   79.6   1   0    0   1     1
## 6   1 319.992  5.10          8.36   79.6   1   0    0   1     1
## 7   1 319.992  7.03          7.47   79.6   1   0    0   1     1
## 8   1 319.992  9.05          6.89   79.6   1   0    0   1     1
## 9   1 319.992 12.12          5.94   79.6   1   0    0   1     1
## 10  1 319.992 24.37          3.28   79.6   1   0    0   1     1
## -----------------------------------
## ----          Model             ----
## -----------------------------------
## Nonlinear mixed-effects model
##   Model function:  One-compartment model with first-order absorption, only CL
##   Model type:  structural
## function(psi,id,xidep) {
##    dose<-xidep[,1]
##    tim<-xidep[,2]
##    ka<-1.5
##    V<-30
##    CL<-psi[id,1]
##    k<-CL/V
##    ypred<-dose*ka/(V*(ka-k))*(exp(-k*tim)-exp(-ka*tim))
##    return(ypred)
## }
## <bytecode: 0x55eb218012e0>
##   Nb of parameters: 1
##       parameter names:  CL
##       distribution:
##      Parameter Distribution Estimated
## [1,] CL        log-normal   Estimated
##   Variance-covariance matrix:
##    CL
## CL  1
##   Error model: constant , initial values: a.1=1
##     No covariate in the model.
##     Initial values
##              CL
## Pop.CondInit 0.5
## ----------------------------------
## ----    Key algorithm options  ----
## ----------------------------------
##     Estimation of individual parameters (MAP)
##     Estimation of standard errors and linearised log-likelihood
##     Estimation of log-likelihood by importance sampling
##     Number of iterations:  K1=300, K2=100
##     Number of chains:  5
##     Seed:  632545
##     Number of MCMC iterations for IS:  5000
##     Simulations:
##         nb of simulated datasets used for npde:  1000
##         nb of simulated datasets used for VPC:  100
##     Input/output
##         save the results to a file:  FALSE
##         save the graphs to files:  FALSE
## ----------------------------------------------------
## ----                  Results                   ----
## ----------------------------------------------------
## -----------------  Fixed effects  ------------------
## ----------------------------------------------------
##      Parameter Estimate SE    CV(%)
## [1,] CL        2.9      0.296 10.3
## [2,] a.1       1.3      0.086  6.8
## ----------------------------------------------------
## -----------  Variance of random effects  -----------
## ----------------------------------------------------
##    Parameter Estimate SE    CV(%)
## CL omega2.CL 0.1      0.052 52
## ----------------------------------------------------
## ------  Correlation matrix of random effects  ------
## ----------------------------------------------------
##           omega2.CL
## omega2.CL 1
## ----------------------------------------------------
## ---------------  Statistical criteria  -------------
## ----------------------------------------------------
## Likelihood computed by linearisation
##       -2LL= 414.8781
##       AIC = 420.8781
##       BIC = 422.3329
##
## Likelihood computed by importance sampling
##       -2LL= 414.4701
##       AIC = 420.4701
##       BIC = 421.9248
## ----------------------------------------------------
```

```
# Old workaround - not needed anymore
if(FALSE) {
  saemix.modeldummypar<-saemixModel(model=model1cpt.onepar,modeltype="structural",
                                    description="One-compartment model with first-order absorption, onl
                                    psi0=matrix(c(0.5,0,0,0),ncol=2,byrow=TRUE, dimnames=list(NULL, c("
                                    transform.par=c(1,0), covariance.model=matrix(c(1,0,0,0),ncol=2), f
  saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE, warnings=FALSE)
  saemix.dummypar<-saemix(saemix.modeldummypar,saemix.data,saemix.options)
}
```

Bootstrap

Continuous data **TODO** warnings on number of covariates in the dataset/model

```
nboot<-5

case.theo <- saemix.bootstrap(theo.fit1, method="case", nboot=nboot)
```

```
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
```

```
cond.theo <- saemix.bootstrap(theo.fit1, method="conditional", nboot=nboot)
```

```
## Simulating data using nsim = 1000 simulated datasets
## Computing WRES and npde ..
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
## Error in dimnames(x) <- dn :
##   la longueur de 'dimnames' [1] n'est pas égale à l'étendue du tableau
```

```
case.theo
```

```
##   Replicate       ka        V        CL beta_Weight(CL)  omega2.ka    omega2.V
## 1         1 1.235957 30.54013 1.3083640     0.009527533 0.35142981 0.013773625
## 2         2 1.190553 32.88848 0.4568100     0.027243508 0.07114035 0.020207059
## 3         3 1.505895 32.14733 2.2705268     0.002532468 0.08565367 0.010826094
## 4         4 1.652374 30.89295 1.3768080     0.009530143 0.54353357 0.009503706
## 5         5 1.335472 32.13102 0.7564938     0.019419208 0.15961123 0.025020251
##     omega2.CL       a.1
## 1 0.104713107 0.7824559
## 2 0.007475325 0.8002388
## 3 0.113405499 0.8424796
## 4 0.063605026 0.7577697
## 5 0.002440983 0.8901749
```

```
cond.theo
```

```
##   Replicate       ka        V        CL beta_Weight(CL) omega2.ka    omega2.V
## 1         1 1.942180 32.17059 0.5844938     0.020807315 0.6297696 0.027472211
## 2         2 1.538108 31.22528 2.2958317     0.001440686 0.1585599 0.008427021
## 3         3 1.320875 34.63802 4.4145093    -0.006255488 0.2425551 0.015612024
## 4         4 1.499881 33.18306 1.4004083     0.009370506 0.1347716 0.033870779
## 5         5 1.742560 31.86650 1.5026058     0.009114581 0.6992376 0.012179287
##    omega2.CL       a.1
## 1 0.07091339 0.7479575
## 2 0.04673639 0.6209496
## 3 0.01675636 0.8586791
## 4 0.03964668 0.8302612
## 5 0.03797403 0.5905750
```

```
nboot<-5

case.count <- saemix.bootstrap(count.fit, method="case", nboot=nboot)
```

```
cond.count <- saemix.bootstrap(count.fit, method="conditional", nboot=nboot)
```

Discrete data

```
## Simulating data using nsim = 1000 simulated datasets
```

```
case.count
```

```
##   Replicate    alpha       beta beta_risk(beta) omega2.alpha omega2.beta
## 1         1 41.50062 0.04197490     -0.09592961   0.06561445  0.04984068
## 2         2 41.67981 0.04433215     -0.23207856   0.05103140  0.04659145
## 3         3 39.84584 0.04269979     -0.32274631   0.06712020  0.03862242
## 4         4 39.92501 0.04293873     -0.15676316   0.09272051  0.03766009
## 5         5 38.59768 0.04472019     -0.22293033   0.05946782  0.04232880
```

```
cond.count
```

```
##   Replicate    alpha       beta beta_risk(beta) omega2.alpha omega2.beta
## 1         1 40.15720 0.03890494    -0.023068965   0.03840495  0.04888234
## 2         2 39.16495 0.03751245     0.006396685   0.04301963  0.04657063
## 3         3 38.23083 0.04654405    -0.133303561   0.05875463  0.07384024
## 4         4 37.38710 0.04301088    -0.163066344   0.03508964  0.05242590
## 5         5 41.23576 0.03987782    -0.057032235   0.05744767  0.05575799
```

BUG

- bug when running the count example with species as a covariate on a parameter (also with a ZIP model with 2 parameters so unrelated to previous problem)

Error in solve.default(comega[Uargs$ind.fix1, Uargs$ind.fix1], rowSums(temp)) : routine Lapack dgesv : le système est exactement singulier : U[2,2] = 0

- **TODO** secure code when mismatch between the number of covariates in the model and the number of covariates in the dataset

```
library(glmmTMB)
data(Salamanders)
summary(Salamanders)
```

```
##       site     mined         cover              sample          DOP         
##  R-1    : 28   yes:308   Min.   :-1.59152   Min.   :1.00   Min.   :-2.1984  
##  R-2    : 28   no :336   1st Qu.:-0.69629   1st Qu.:1.75   1st Qu.:-0.3018  
##  R-3    : 28             Median :-0.04974   Median :2.50   Median :-0.0916  
##  R-4    : 28             Mean   : 0.00000   Mean   :2.50   Mean   : 0.0000  
##  R-5    : 28             3rd Qu.: 0.59682   3rd Qu.:3.25   3rd Qu.: 0.0000  
##  R-6    : 28             Max.   : 1.88993   Max.   :4.00   Max.   : 3.1691  
##  (Other):476                                                                
##      Wtemp               DOY               spp         count       
##  Min.   :-3.0234   Min.   :-2.7122   GP   :92   Min.   : 0.000  
##  1st Qu.:-0.6139   1st Qu.:-0.5653   PR   :92   1st Qu.: 0.000  
##  Median : 0.0370   Median :-0.0590   DM   :92   Median : 0.000  
##  Mean   : 0.0000   Mean   : 0.0000   EC-A :92   Mean   : 1.323  
##  3rd Qu.: 0.6032   3rd Qu.: 0.9739   EC-L :92   3rd Qu.: 2.000  
##  Max.   : 2.2094   Max.   : 1.4600   DES-L:92   Max.   :36.000  
##                                      DF   :92
```

```
twospecies<-Salamanders[Salamanders$spp %in% c("EC-L","DF"),]
twospecies$spp <- as.character(twospecies$spp)
```

```
saemix.data<-saemixData(name.data=twospecies, name.group=c("site"),
                        name.predictors=c("DOY","count"),name.response=c("count"),
                        name.covariates=c("spp","mined","cover", "DOP", "Wtemp"),
                        units=list(x="day (scaled)",y="",covariates=c("","","","","")))
```

```
##
##
## The following SaemixData object was successfully created:
##
## Object of class SaemixData
##     longitudinal data for use with the SAEM algorithm
## Dataset twospecies
##     Structured data: count ~ DOY + count | site
##     X variable for graphs: DOY (day (scaled))
##     covariates: spp (), mined (), cover (), DOP (), Wtemp ()
##       reference class for covariate spp :  DF
##       reference class for covariate mined :  yes
```

```
countmodel.poisson<-function(psi,id,xidep) {
  y<-xidep[,2]
  lambda<-psi[id,1]
  logp <- -lambda + y*log(lambda) - log(factorial(y))
  return(logp)
}

saemix.model<-saemixModel(model=countmodel.poisson,description="count model Poisson",modeltype="likelih
                          psi0=matrix(c(1),ncol=1,byrow=TRUE,dimnames=list(NULL, c("lambda"))),
                          covariate.model = matrix(c(1,rep(0,4)),ncol=1, byrow=T),
                          transform.par=c(1)) #omega.init=matrix(c(0.5,0,0,0.3),ncol=2,byrow=TRUE))
```

```
##
##
## The following SaemixModel object was successfully created:
##
## Nonlinear mixed-effects model
##   Model function:  count model Poisson
##   Model type:  likelihood
## function(psi,id,xidep) {
##   y<-xidep[,2]
##   lambda<-psi[id,1]
##   logp <- -lambda + y*log(lambda) - log(factorial(y))
##   return(logp)
## }
##   Nb of parameters: 1
##       parameter names:  lambda
##       distribution:
##      Parameter Distribution Estimated
## [1,] lambda    log-normal   Estimated
##   Variance-covariance matrix:
##        lambda
## lambda      1
##   Covariate model:
##      lambda
## [1,]      1
## [2,]      0
## [3,]      0
## [4,]      0
## [5,]      0
##     Initial values
##              lambda
## Pop.CondInit      1
## Cov.CondInit      0
```

```
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE, displayProgress=FALSE, fim=FALSE)
poisson.fit<-try(saemix(saemix.model,saemix.data,saemix.options)) # fails, pb with solving Lapack
```

```
## Error in solve.default(comega[Uargs$ind.fix1, Uargs$ind.fix1], rowSums(temp)) :
##   Routine Lapack dgesv : le système est exactement singulier : U[2,2] = 0
```