Package 'saemix'

July 1, 2011

Type Package

Title Stochastic Approximation Expectation Maximization (SAEM) algorithm

Version 0.96

Date 2011-04-07

Author Emmanuelle Comets, Audrey Lavenu, Marc Lavielle.

Maintainer Emmanuelle Comets <emmanuelle.comets@inserm.fr>

Description The SAEM package implements the Stochastic Approximation EM algorithm for parameter estimation in (non)linear mixed effects models. The SAEM algorithm: - computes the maximum likelihood estimator of the population parameters, without any approximation of the model (linearization, quadrature approximation,...), using the Stochastic Approximation Expectation Maximization (SAEM) algorithm, - provides standard errors for the maximum likelihood estimator - estimates the conditional modes, the conditional means and the conditional standard deviations of the individual parameters, using the Hastings-Metropolis algorithm. Several applications of SAEM in agronomy, animal breeding and PKPD analysis have been published by members of the Monolix group (http://software.monolix.org/).

License GPL (>= 2)

LazyLoad yes

LazyData yes

Depends methods

Imports graphics, stats

Collate global.R SaemixData.R SaemixModel.R SaemixRes.R SaemixObject.R main.R zzz.R

R topics documented:

`saemix-package` *Stochastic Approximation Expectation Maximization (SAEM) algorithm for non-linear mixed effects models*

Description

- Computing the maximum likelihood estimator of the population parameters, without any approximation of the model (linearization, quadrature approximation, . . .), using the Stochastic Approximation Expectation Maximization (SAEM) algorithm
- Estimation of the Fisher Information matrix
- Estimation of the individual parameters
- Estimation of the likelihood
- Plot convergence graphs

Details

Package:	saemix
Type:	Package
Version:	0.9
Date:	2010-09-19
License:	GPL (>=) 1.2
LazyLoad:	yes

The SAEM package includes a number of undocumented functions, which are not meant to be used directly by the user.

default setdefault

computational functions cutoff,cutoff.max, cutoff.eps, cutoff.res, compute.Uy, compute.Uy.nocov, conditional.distribution, gqg.mlx

distributions normcdf, norminv

error model error

sampling trnd.mlx, tpdf.mlx, gammarnd.mlx

parameter transformations transpsi, transphi, dtransphi

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038. Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

nlme,SaemixData,SaemixModel, SaemixObject,saemix

Examples

```
data(theo.saemix)

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")

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.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE), error.model="constant")

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE)

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

print(saemix.fit)

plot(saemix.fit)
```

`coef-methods` *Methods for Function coef*

Description

Methods for function `coef`

Methods

`signature(x = "ANY")` default coef function ?

`signature(x = "SaemixObject")` extracts coefficients from an SaemixObject

`conddist.saemix` *Estimate conditional mean and variance of individual parameters using the MCMC algorithm*

Description

When the parameters of the model have been estimated, we can estimate the individual parameters (psi_i).

Let hattheta be the estimated value of theta computed with the SAEM algorithm and let p(phi_i |y_i; hattheta) be the conditional distribution of phi_i for 1<=i<=N . We use the MCMC procedure used in the SAEM algorithm to estimate these conditional distributions. We empirically estimate the conditional mean E(phi_i |y_i; hattheta) and the conditional standard deviation sd(phi_i |y_i; hattheta).

Usage

```
conddist.saemix(saemixObject,nsamp=1,max.iter=NULL,...)
```

Arguments

saemixObject	an object returned by the saemix function
nsamp	Number of samples to be drawn in the conditional distribution for each subject. Defaults to 1
max.iter	Maximum number of iterations for the computation of the conditional estimates. Defaults to twice the total number of iterations (sum(saemixObject["options"]$nbiter.saemix)*2)
...	optional arguments passed to the plots. Plots will appear if the option displayProgress in the saemixObject object is TRUE

Details

See PDF documentation for details of the computation. Briefly, the MCMC algorithm is used to obtain samples from the individual conditional distributions of the parameters. The algorithm is initialised for each subject to the conditional estimate of the individual parameters obtained at the end of the SAEMIX fit. A convergence criterion is used to ensure convergence of the mean and variance of the conditional distributions. When nsamp>1, several chains of the MCMC algorithm are run in parallel to obtain samples from the conditional distributions, and the convergence criterion must be achieved for all chains. When nsamp>1, the estimate of the conditional mean is obtained by averaging over the different samples.

The shrinkage for any given parameter for the conditional estimate is obtained as

Sh=1-var(eta_i)/omega(eta)

where var(eta_i) is the empirical variance of the estimates of the individual random effects, and omega(eta) is the estimated variance.

The function adds or modifies the following elements in the results:

cond.mean.phi Conditional mean of the individual distribution of the parameters (obtained as the mean of the samples)

cond.var.phi Conditional variance of the individual distribution of the parameters (obtained as the mean of the estimated variance of the samples)

cond.shrinkage Estimate of the shrinkage for the conditional estimates

cond.mean.eta Conditional mean of the individual distribution of the parameters (obtained as the mean of the samples)

phi.samp An array with 3 dimensions, giving nsamp samples from the conditional distributions of the individual parameters

phi.samp.var The estimated individual variances for the sampled parameters phi.samp

A warning is output if the maximum number of iterations is reached without convergence (the maximum number of iterations is saemix.options$nbiter.saemix[2]).

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

SaemixData,SaemixModel, SaemixObject,saemixControl,saemix

Examples

```
data(theo.saemix)

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")

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.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE), error.model="constant")

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE)

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

saemix.fit<-conddist.saemix(saemix.fit,nsamp=3)
# First sample from the conditional distribution
# (a N (nb of subject) by nb.etas (nb of parameters) matrix)
saemix.fit["results"]["phi.samp"][,,1]

# Second sample
saemix.fit["results"]["phi.samp"][,,2]
```

---

cow.saemix *Evolution of the weight of 560 cows, in SAEM format*

Description

cow.saemix contains data from winter wheat experiments.

Usage

```
cow.saemix
```

Format

This data frame contains the following columns:

cow: the id.

time: time (days).

weight: weight of the cow (kg).

birthyear: year of birth (between 1988 and 1998).

twin: existence of a twin (no=1, yes=2).

birthrank: the rank of birth (beetween 3 and 7).

Details

The data used in this example is the evolution of the weight (in kg) of 560 cows. We have 9 or 10 measurements per subject.

We use an exponential growth model for this data: y_ij = A_i (1- B_i exp(- K_i t_ij)) +epsilon_ij

Examples

```
data(cow.saemix)
saemix.data<-saemixData(name.data=cow.saemix,header=TRUE,name.group=c("cow"),
  name.predictors=c("time"),name.response=c("weight"),
  name.covariates=c("birthyear","twin","birthrank"),
  units=list(x="days",y="kg",covariates=c("yr","-","-")))

growthcow<-function(psi,id,xidep) {
# input:
#   psi : matrix of parameters (3 columns, a, b, k)
#   id : vector of indices
#   xidep : dependent variables (same nb of rows as length of id)
# returns:
#   a vector of predictions of length equal to length of id
  x<-xidep[,1]
  a<-psi[id,1]
  b<-psi[id,2]
  k<-psi[id,3]
  f<-a*(1-b*exp(-k*x))
  return(f)
}
saemix.model<-saemixModel(model=growthcow,
  description="Exponential growth model",
  psi0=matrix(c(700,0.9,0.02,0,0,0),ncol=3,byrow=TRUE,
  dimnames=list(NULL,c("A","B","k"))),transform.par=c(1,1,1),fixed.estim=c(1,1,1),
  covariate.model=matrix(c(0,0,0),ncol=3,byrow=TRUE),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="constant")

saemix.options<-list(algorithms=c(1,1,1),nb.chains=1,nbiter.saemix=c(200,100),
  seed=4526,save=FALSE,save.graphs=FALSE)
```

```
# Plotting the data
plot(saemix.data,xlab="Time (day)",ylab="Weight of the cow (kg)")

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

`default.saemix.plots` *Wrapper functions to produce certain sets of default plots*

Description

These functions produce default sets of plots, corresponding to diagnostic or individual fits.

Usage

```
default.saemix.plots(saemixObject, ...)
basic.gof(saemixObject, ...)
advanced.gof(saemixObject, ...)
covariate.fits(saemixObject, which = "parameters", ...)
individual.fits(saemixObject, ...)
```

Arguments

`saemixObject`	an object returned by the `saemix` function
`which`	for covariate fits, whether they should be produced with the EBE estimates of the parameters (the default) or with random effects (which="randeff")
`...`	optional arguments passed to the plots

Details

These functions are wrapper functions designed to produce default sets of plots to help the user assess their model fits.

Value

`default.saemix.plots`
: by default, the following plots are produced: a plot of the data, convergence plots, plot of the likelihood by importance sampling (if computed), plots of observations versus predictions, scatterplots and distribution of residuals, boxplot of the random effects, correlations between random effects, distribution of the parameters, VPC

`basic.gof`
: basic goodness-of-fit plots: convergence plots, plot of the likelihood by importance sampling (if computed), plots of observations versus predictions

`advanced.gof`
: advanced goodness-of-fit plots: scatterplots and distribution of residuals, VPC,...

`covariate.fits`
: plots of all estimated parameters versus all covariates in the dataset

`individual.fits`
: plots of individual predictions (line) overlayed on individual observations (dots) for all subjects in the dataset

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

saemix, saemix.plot.data, saemix.plot.setoptions, plot.saemix

Examples

```
data(theo.saemix)

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")

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.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="constant")

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE)

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

default.saemix.plots(saemix.fit)

basic.gof(saemix.fit)

advanced.gof(saemix.fit)

individual.fits(saemix.fit)
```

`fim.saemix` *Computes the Fisher Information Matrix by linearisation*

Description

Estimate by linearisation the Fisher Information Matrix and the standard error of the estimated parameters.

Usage

```
fim.saemix(saemixObject)
```

Arguments

`saemixObject` an object returned by the `saemix` function

Details

The inverse of the Fisher Information Matrix provides an estimate of the variance of the estimated parameters theta. This matrix cannot be computed in closed-form for nonlinear mixed-effect models; instead, an approximation is obtained as the Fisher Information Matrix of the Gaussian model deduced from the nonlinear mixed effects model after linearisation of the function f around the conditional expectation of the individual Gaussian parameters. This matrix is a block matrix (no correlations between the estimated fixed effects and the estimated variances).

Value

The function returns an updated version of the object saemix.fit in which the following elements have been added:

se.fixed: standard error of fixed effects, obtained as part of the diagonal of the inverse of the Fisher Information Matrix (only when fim.saemix has been run, or when the saemix.options$algorithms[2] is 1)

se.omega2: standard error of the variance of random effects, obtained as part of the diagonal of the inverse of the Fisher Information Matrix (only when fim.saemix has been run, or when the saemix.options$algorithms[2] is 1)

se.res: standard error of the parameters of the residual error model, obtained as part of the diagonal of the inverse of the Fisher Information Matrix (only when fim.saemix has been run, or when the saemix.options$algorithms[2] is 1)

fim: Fisher Information Matrix

ll.lin: likelihood calculated by linearisation

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

SaemixObject,saemix

Examples

```
# Running the main algorithm to estimate the population parameters
data(theo.saemix)

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")

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.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE), error.model="constant")

saemix.options<-list(algorithm=c(1,0,0),seed=632545,save=FALSE,save.graphs=FALSE)

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

# Estimating the Fisher Information Matrix using the result of saemix
# & returning the result in the same object

fim.saemix(saemix.fit)
```

---

initialize-methods *Methods for Function initialize*

Description

Constructor functions for Classes in the saemix package (not user-level)

Methods

`signature(.Object = "SaemixData")` create a SaemixData object. Please use the saemixData function.

`signature(.Object = "SaemixModel")` create a SaemixModel object Please use the `saemixModel` function.

`signature(.Object = "SaemixObject")` create a SaemixObject object. This object is obtained after a successful call to `saemix`

`signature(.Object = "SaemixRepData")` create a SaemixRepData object

`signature(.Object = "SaemixRes")` create a SaemixRes object

`signature(.Object = "SaemixSimData")` create a SaemixSimData object

`llgq.saemix` *Log-likelihood using Gaussian Quadrature*

Description

Estimate the log-likelihood using Gaussian Quadrature (multidimensional grid)

Usage

```
llgq.saemix(saemixObject)
```

Arguments

`saemixObject` an object returned by the `saemix` function

Details

The likelihood of the observations is estimated using Gaussian Quadrature (see documentation).

Value

the log-likelihood estimated by Gaussian Quadrature

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

`SaemixObject`,`saemix`,`llis.saemix`

Examples

```
# Running the main algorithm to estimate the population parameters
data(theo.saemix)
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")

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.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE), error.model="constant")

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE)

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

# Estimating the likelihood by Gaussian Quadrature using the result of saemix
# & returning the result in the same object
saemix.fit<-llgq.saemix(saemix.fit)
```

---

llis.saemix *Log-likelihood using Importance Sampling*

Description

Estimate the log-likelihood using Importance Sampling

Usage

```
llis.saemix(saemixObject)
```

Arguments

`saemixObject`	an object returned by the `saemix` function

Details

The likelihood of the observations is estimated without any approximation using a Monte-Carlo approach (see documentation).

Value

the log-likelihood estimated by Importance Sampling

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

SaemixObject,saemix,llgq.saemix

Examples

```
# Running the main algorithm to estimate the population parameters
data(theo.saemix)
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")

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.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="constant")

saemix.options<-list(algorithm=c(1,0,0),seed=632545,save=FALSE,save.graphs=FALSE)

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

# Estimating the likelihood by importance sampling using the result of saemix
# & returning the result in the same object
saemix.fit<-llis.saemix(saemix.fit)
```

`map.saemix` *Estimates of the individual parameters (conditional mode)*

Description

Compute the estimates of the individual parameters PSI_i (conditional mode - Maximum A Posteriori)

Usage

```
map.saemix(saemixObject)
```

Arguments

`saemixObject`	an object returned by the `saemix` function

Details

The MCMC procedure is used to estimate the conditional mode (or Maximum A Posteriori) m(phi_i |yi ; hattheta) = Argmax_phi_i p(phi_i |yi ; hattheta)

Value

`saemixObject`:
returns the object with the estimates of the MAP parameters (see example for usage)

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

`SaemixObject`,`saemix`

Examples

```
data(theo.saemix)

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")

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.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="constant")

saemix.options<-list(algorithm=c(1,0,0),seed=632545,
  save=FALSE,save.graphs=FALSE)

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

# Estimating the individual parameters using the result of saemix
# & returning the result in the same object

saemix.fit<-map.saemix(saemix.fit)
```

`oxboys.saemix` *Heights of Boys in Oxford*

Description

The `oxboys.saemix` data frame has 234 rows and 4 columns.

Usage

```
oxboys.saemix
```

Format

This data frame contains the following columns:

Subject: an ordered factor giving a unique identifier for each boy in the experiment

age: a numeric vector giving the standardized age (dimensionless)

height: a numeric vector giving the height of the boy (cm)

Occasion: an ordered factor - the result of converting 'age' from a continuous variable to a count so these slightly unbalanced data can be analyzed as balanced.

Details

These data are described in Goldstein (1987) as data on the height of a selection of boys from Oxford, England versus a standardized age. The dataset can be found in the package `nlme`.

We use an linear model for this data: y_ij = Base_i + slope_i x_ij +epsilon_ij

Source

Pinheiro, J. C. and Bates, D. M. (2000), _Mixed-Effects Models in S and S-PLUS_, Springer, New York. (Appendix A.19)

Examples

```
data(oxboys.saemix)
saemix.data<-saemixData(name.data=oxboys.saemix,header=TRUE,
  name.group=c("Subject"),name.predictors=c("age"),name.response=c("height"),
  units=list(x="yr",y="cm"))

growth.linear<-function(psi,id,xidep) {
# input:
#   psi : matrix of parameters (2 columns, base and slope)
#   id : vector of indices
#   xidep : dependent variables (same nb of rows as length of id)
# returns:
#   a vector of predictions of length equal to length of id
  x<-xidep[,1]
  base<-psi[id,1]
  slope<-psi[id,2]
  f<-base+slope*x
  return(f)
}
saemix.model<-saemixModel(model=growth.linear,description="Linear model",
  psi0=matrix(c(140,1),ncol=2,byrow=TRUE,dimnames=list(NULL,c("base","slope"))),
  transform.par=c(1,0),covariance.model=matrix(c(1,1,1,1),ncol=2,byrow=TRUE),
  error.model="constant")

saemix.options<-list(algorithms=c(1,1,1),nb.chains=1,seed=201004,
  save=FALSE,save.graphs=FALSE)

# plot the data
plot(saemix.data)

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

PD1.saemix — *Data simulated according to an Emax response model, in SAEM format*

Description

`PD1.saemix` contains data from winter wheat experiments.

Usage

```
PD1.saemix

PD2.saemix
```

Format

This data frame contains the following columns:

subject: the site number

dose: simulated dose.

response: simulated response.

gender: gender (0 for male, 1 for female).

Details

These examples were used by P. Girard and F. Mentre for the symposium dedicated to Comparison of Algorithms Using Simulated Data Sets and Blind Analysis, that took place in Lyon, France, September 2004.

The dataset contains 100 individuals, each receiving 3 different doses:(0, 10, 90), (5, 25, 65) or (0, 20, 30). It was assumed that doses were given in a cross-over study with sufficient wash out period to avoid carry over. Responses (y_ij) were simulated with the following pharmacodynamic model:

y_ij = E0_i + D_ij Emax_i/(D_ij + ED50_i) +epsilon_ij

The individual parameters were simulated according to

log (E0_i) = log (E0) + eta_i1 log (Emax_i) = log (Emax) + eta_i2 log (E50_i) = log (E50) + beta w_i + eta_i3

PD1.saemix contains the data simulated with a gender effect, beta=0.3.

PD2.saemix contains the data simulated without a gender effect, beta=0.

Source

Girard P., Mentre F. Comparison of Algorithms Using Simulated Data Sets and Blind Analysis workshop, Lyon, France, September 2004.

Examples

```
data(PD1.saemix)
saemix.data<-saemixData(name.data=PD1.saemix,header=TRUE,name.group=c("subject"),
  name.predictors=c("dose"),name.response=c("response"),
  name.covariates=c("gender"), units=list(x="mg",y="-",covariates=c("-")))

modelemax<-function(psi,id,xidep) {
# input:
#   psi : matrix of parameters (3 columns, E0, Emax, EC50)
#   id : vector of indices
#   xidep : dependent variables (same nb of rows as length of id)
# returns:
#   a vector of predictions of length equal to length of id
  dose<-xidep[,1]
  e0<-psi[id,1]
  emax<-psi[id,2]
  e50<-psi[id,3]
  f<-e0+emax*dose/(e50+dose)
  return(f)
}
saemix.model<-saemixModel(model=modelemax,description="Emax growth model",
```

```
  psi0=matrix(c(20,300,20,0,0,0),ncol=3,byrow=TRUE,
  dimnames=list(NULL, c("E0","Emax","EC50"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,0,1),ncol=3,byrow=TRUE),
  fixed.estim=c(1,1,1),covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),
  ncol=3,byrow=TRUE),error.model="constant")

saemix.options<-list(algorithms=c(1,1,1),nb.chains=1,seed=42,
  save=FALSE,save.graphs=FALSE)

# Plotting the data
plot(saemix.data,main="Simulated data PD1")

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

# Compare models with and without covariates with LL by Importance Sampling
# SE not computed
model1<-saemixModel(model=modelemax,description="Emax growth model",
  psi0=matrix(c(20,300,20,0,0,0),ncol=3,byrow=TRUE,dimnames=list(NULL,
  c("E0","Emax","EC50"))), transform.par=c(1,1,1),
  covariate.model=matrix(c(0,0,0), ncol=3,byrow=TRUE),fixed.estim=c(1,1,1))

model2<-saemixModel(model=modelemax,description="Emax growth model",
  psi0=matrix(c(20,300,20,0,0,0),ncol=3,byrow=TRUE,dimnames=list(NULL,
  c("E0","Emax","EC50"))), transform.par=c(1,1,1),
  covariate.model=matrix(c(0,0,1), ncol=3,byrow=TRUE),fixed.estim=c(1,1,1))

saemix.options<-list(algorithms=c(0,1,1),nb.chains=3,seed=765754,
  nbiter.saemix=c(500,300),save=FALSE,save.graphs=FALSE)

fit1<-saemix(model1,saemix.data,saemix.options)
fit2<-saemix(model2,saemix.data,saemix.options)
wstat<-(-2)*(fit1["results"]["ll.is"]-fit2["results"]["ll.is"])

cat("LRT test for covariate effect on EC50: p-value=",1-pchisq(wstat,1),"\n")
```

`plot-methods` *Methods for Function plot*

Description

Methods for function `plot`

Methods

`signature(x = "ANY")` default plot function ?

`signature(x = "SaemixData")` Plots the data. Defaults to a spaghetti plot of response versus predictor, with lines joining the data for one individual

`signature(x = "SaemixModel")` Plots prediction of the model

`signature(x = "SaemixObject")` This method gives access to a number of plots that can be performed on a SaemixObject

`signature(x = "SaemixSimData")` Plots simulated datasets

predict-methods *Methods for Function predict*

Description

Methods for function `predict`

Methods

`signature(object = "ANY")` Default predict functions

`signature(object = "SaemixObject")` Computes predictions using the results of an SAEM fit

print-methods *Methods for Function print*

Description

Prints a summary of an object

Methods

`signature(x = "ANY")` Default print function

`signature(x = "SaemixData")` Prints a summary of a SaemixData object

`signature(x = "SaemixModel")` Prints a summary of a SaemixModel object

`signature(x = "SaemixObject")` Prints a summary of the results from a SAEMIX fit

`signature(x = "SaemixRes")` Not user-level

psi *Functions to extract the individual estimates of the parameters and random effects*

Description

These three functions are used to access the estimates of individual parameters and random effects.

Usage

```
psi(object, indiv.par)
phi(object, indiv.par)
eta(object, indiv.par)
```

Arguments

`object`	an object returned by the `saemix` function
`indiv.par`	a string giving the type of estimate to be used (one of "map", for the Maximum A Posteriori estimate, or "eap", for conditional estimate). Defaults to "map"

Details

The psi_i represent the individual parameter estimates. In the SAEM algorithm, these parameters are assumed to be a transformation of a Gaussian random vector phi_i, where the phi_i can be written as a function of the individual random effects (eta_i), the covariate matrix (C_i) and the vector of fixed effects (mu):

phi_i = C_i mu + eta_i

More details can be found in the PDF documentation.

Value

These functions are used to access and output the estimates of parameters and random effects. When the object passed to the function does not contain these estimates, they are automatically computed. The object is then returned (invisibly) with these estimates added to the results.

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

SaemixData,SaemixModel, SaemixObject, saemixControl, plot.saemix

Examples

```
data(theo.saemix)

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")

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.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
```

```
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="constant")

saemix.options<-list(algorithm=c(1,0,0),seed=632545,save=FALSE,save.graphs=FALSE)

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

psi(saemix.fit)
phi(saemix.fit)
eta(saemix.fit,indiv.par="eap")
```

`psi-methods` *Methods for Functions psi, phi and eta*

Description

These methods are used to access estimates of individual parameters and random effects

Methods

`signature(object = "SaemixObject")` please refer to the PDF documentation for the models

`saemix` *Stochastic Approximation Expectation Maximization (SAEM) algorithm*

Description

SAEM algorithm perform parameter estimation for nonlinear mixed effects models without any approximation of the model (linearization, quadrature approximation, . . .)

Usage

```
saemix(model, data, control = list())
```

Arguments

`model`	an object of class SaemixModel, created by a call to the function `saemixModel`
`data`	an object of class SaemixData, created by a call to the function `saemixData`
`control`	a list of options, see `saemixControl`

Details

The SAEM algorithm is a stochastic approximation version of the standard EM algorithm proposed by Kuhn and Lavielle (see reference). Details of the algorithm can be found in the pdf file accompanying the package.

Value

An object of class SaemixObject containing the results of the fit of the data by the non-linear mixed effect model. A summary of the results is printed out to the terminal, and, provided the appropriate options have not been changed, numerical and graphical outputs are saved in a directory.

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

SaemixData,SaemixModel, SaemixObject, saemixControl, plot.saemix

Examples

```
data(theo.saemix)

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")

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.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="constant")

saemix.fit<-saemix(saemix.model,saemix.data,list(seed=632545,directory="newtheo",
  save=FALSE,save.graphs=FALSE))

# Prints a summary of the results
print(saemix.fit)

# Outputs the estimates of individual parameters
psi(saemix.fit)
```

```
# Shows some diagnostic plots to evaluate the fit
plot(saemix.fit)
```

`saemix.plot.data` *Functions implementing each type of plot in SAEM*

Description

Several plots (selectable by the type argument) are currently available: convergence plot, individual plots, predictions versus observations, distribution plots, VPC, residual plots.

Usage

```
saemix.plot.data(saemixObject, ...)
saemix.plot.convergence(saemixObject,niter=0, ...)
saemix.plot.llis(saemixObject, ...)
saemix.plot.obsvspred(saemixObject, ...)
saemix.plot.distribresiduals(saemixObject, ...)
saemix.plot.scatterresiduals(saemixObject, ...)
saemix.plot.fits(saemixObject, ...)
saemix.plot.distpsi(saemixObject, ...)
saemix.plot.randeff(saemixObject, ...)
saemix.plot.correlations(saemixObject, ...)
saemix.plot.parcov(saemixObject, ...)
saemix.plot.randeffcov(saemixObject, ...)
saemix.plot.npde(saemixObject, ...)
saemix.plot.vpc(saemixObject,npc = FALSE, ...)

saemix.plot.parcov.aux(saemixObject, partype = "p", ...)
compute.sres(saemixObject)
compute.eta.map(saemixObject)
```

Arguments

`saemixObject`	an object returned by the `saemix` function
`...`	optional arguments passed to the plots
`npc`	for VPC, computes Numerical Predictive Checks (currently not implemented)
`niter`	the convergence plots are shown up to iteration "niter". Defaults to 0, which indicates the convergence plots should be plotted up to the maximal number of iterations set for the algorithm
`partype`	(this function is not user-level)

Details

These functions implement plots different graphs related to the algorithm (convergence plots, likelihood estimation) as well as diagnostic graphs. A description is provided in the PDF documentation.

saemix.plot.parcov.aux, compute.sres and compute.eta.map are helper functions, not intended to be called by the user directly.

By default, the following plots are produced:

saemix.plot.data: A spaghetti plot of the data, displaying the observed data y as a function of the regression variable (time for a PK application)

saemix.plot.convergence: For each parameter in the model, this plot shows the evolution of the parameter estimate versus the iteration number

saemix.plot.llis: Graph showing the evolution of the log-likelihood during the estimation by importance sampling

saemix.plot.obsvspred: Plot of the predictions computed with the population parameters versus the observations (left), and plot of the predictions computed with the individual parameters versus the observations (right)

saemix.plot.scatterresiduals: Scatterplot of the residuals versus the predictor (top) and versus predictions (bottom), for weighted residuals (population residuals, left), individual weighted residuals (middle) and npde (right).

saemix.plot.distribresiduals: Distribution of the residuals, plotted as histogram (top) and as a QQ-plot (bottom), for weighted residuals (population residuals, left), individual weighted residuals (middle) and npde (right).

saemix.plot.fits: Model fits. Individual fits are obtained using the individual parameters with the individual covariates. Population fits are obtained using the population parameters with the individual covariates (red) and the individual parameters with the individual covariates (green). By default the individual plots are displayed.

saemix.plot.distpsi: Distribution of the parameters (conditional on covariates when some are included in the model). A histogram of individual parameter estimates can be overlayed on the plot, but it should be noted that the histogram does not make sense when there are covariates influencing the parameters and a warning will be displayed

saemix.plot.randeff: Boxplot of the random effects

saemix.plot.correlations: Correlation between the random effects

saemix.plot.parcov: Plots of the estimates of the individual parameters versus the covariates, using scatterplot for continuous covariates, boxplot for categorical covariates

saemix.plot.randeffcov: Plots of the estimates of the random effects versus the covariates, using scatterplot for continuous covariates, boxplot for categorical covariates

saemix.plot.npde: Plots 4 graphs to evaluate the shape of the distribution of the normalised prediction distribution errors (npde)

saemix.plot.vpc: Visual Predictive Check, with options to include the prediction intervals around the boundaries of the selected interval as well as around the median (50th percentile of the simulated data). Several methods are available to define binning on the X-axis (see methods in the PDF guide).

Each plot can be customised by modifying options, either through a list of options set by the `saemix.plot.setoptions` function, or on the fly by passing an option in the call to the plot (see examples).

Value

None

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

SaemixObject,saemix,saemix.plot.setoptions,saemix.plot.select,plot.saemix

Examples

```
data(theo.saemix)

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")

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.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="constant")

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE)

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

# Simulate data and compute weighted residuals and npde
saemix.fit<-compute.sres(saemix.fit)

# Data
saemix.plot.data(saemix.fit)

# Convergence
saemix.plot.convergence(saemix.fit)

# Individual plot for subject 1, smoothed
saemix.plot.fits(saemix.fit,ilist=1,smooth=TRUE)

# Individual plot for subject 1 to 12, with ask set to TRUE
# (the system will pause before a new graph is produced)
saemix.plot.fits(saemix.fit,ilist=c(1:12),ask=TRUE)
```

```
# Diagnostic plot: observations versus population predictions
par(mfrow=c(1,1))
saemix.plot.obsvspred(saemix.fit,level=0,new=FALSE)

# LL by Importance Sampling
saemix.plot.llis(saemix.fit)

# Scatter plot of residuals
saemix.plot.scatterresiduals(saemix.fit)

# Boxplot of random effects
saemix.plot.randeff(saemix.fit)

# Relationships between parameters and covariates
saemix.plot.parcov(saemix.fit)

# Relationships between parameters and covariates, on the same page
par(mfrow=c(3,2))
saemix.plot.parcov(saemix.fit,new=FALSE)

# VPC, default options (10 bins, equal number of observations in each bin)
saemix.plot.vpc(saemix.fit)

# VPC, user-defined breaks for binning
saemix.plot.vpc(saemix.fit,vpc.method="user",
  vpc.breaks=c(0.4,0.8,1.5,2.5,4,5.5,8,10,13))
```

---

saemix.plot.select *Plots of the results obtained by SAEM*

Description

Several plots (selectable by the type argument) are currently available: convergence plot, individual plots, predictions versus observations, distribution plots, residual plots, VPC.

Usage

```
saemix.plot.select(saemixObject, data = FALSE, convergence = FALSE,
  likelihood = FALSE, individual.fit = FALSE, population.fit = FALSE,
  both.fit = FALSE, observations.vs.predictions = FALSE,
  residuals.scatter = FALSE, residuals.distribution = FALSE,
  random.effects = FALSE, correlations = FALSE,
  parameters.vs.covariates = FALSE, randeff.vs.covariates = FALSE,
  marginal.distribution = FALSE, vpc = FALSE, npde = FALSE, ...)
```

Arguments

saemixObject	an object returned by the saemix function
data	if TRUE, produce a plot of the data. Defaults to FALSE
convergence	if TRUE, produce a convergence plot. Defaults to FALSE
likelihood	if TRUE, produce a plot of the estimation of the LL by importance sampling. Defaults to FALSE

`individual.fit`
: if TRUE, produce individual fits with individual estimates. Defaults to FALSE

`population.fit`
: if TRUE, produce individual fits with population estimates. Defaults to FALSE

`both.fit`
: if TRUE, produce individual fits with both individual and population estimates. Defaults to FALSE

`observations.vs.predictions`
: if TRUE, produce a plot of observations versus predictions. Defaults to FALSE

`residuals.scatter`
: if TRUE, produce scatterplots of residuals versus predictor and predictions. Defaults to FALSE

`residuals.distribution`
: if TRUE, produce plots of the distribution of residuals. Defaults to FALSE

`random.effects`
: if TRUE, produce boxplots of the random effects. Defaults to FALSE

`correlations`
: if TRUE, produce a matrix plot showing the correlation between random effects. Defaults to FALSE

`parameters.vs.covariates`
: if TRUE, produce plots of the relationships between parameters and covariates, using the Empirical Bayes Estimates of individual parameters. Defaults to FALSE

`randeff.vs.covariates`
: if TRUE, produce plots of the relationships between random effects and covariates, using the Empirical Bayes Estimates of individual random effects. Defaults to FALSE

`marginal.distribution`
: if TRUE, produce plots of the marginal distribution of the random effects. Defaults to FALSE

`vpc`
: if TRUE, produce Visual Predictive Check plots. Defaults to FALSE

`npde`
: if TRUE, produce plots of the npde. Defaults to FALSE

`...`
: optional arguments passed to the plots

Details

This function plots different graphs related to the algorithm (convergence plots, likelihood estimation) as well as diagnostic graphs. A description is provided in the PDF documentation.

data A spaghetti plot of the data, displaying the observed data y as a function of the regression variable (eg time for a PK application)

convergence For each parameter in the model, this plot shows the evolution of the parameter estimate versus the iteration number

likelihood Estimation of the likelihood estimated by importance sampling, as a function of the number of MCMC samples

individual.fit Individual fits, using the individual parameters with the individual covariates

population.fit Individual fits, using the population parameters with the individual covariates

both.fit Individual fits, using the population parameters with the individual covariates and the individual parameters with the individual covariates

observations.vs.predictions Plot of the predictions computed with the population parameters versus the observations (left), and plot of the predictions computed with the individual parameters versus the observations (right)

residuals.scatter Scatterplot of standardised residuals versus the X predictor and versus predictions. These plots are shown for individual and population residuals, as well as for npde when they are available

residuals.distribution Distribution of standardised residuals, using histograms and QQ-plot. These plots are shown for individual and population residuals, as well as for npde when they are available

random.effects Boxplot of the random effects

correlations Correlation between the random effects, with a smoothing spline

parameters.versus.covariates Plots of the estimate of the individual parameters versus the covariates, using scatterplot for continuous covariates, boxplot for categorical covariates

randeff.versus.covariates Plots of the estimate of the individual random effects versus the covariates, using scatterplot for continuous covariates, boxplot for categorical covariates

marginal.distribution Distribution of each parameter in the model (conditional on covariates when some are included in the model)

npde Plot of npde as in package npde

vpc Visual Predictive Check

Value

None

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

SaemixObject,saemix, default.saemix.plots, saemix.plot.setoptions, saemix.plot.data, saemix.plot.convergence, saemix.plot.llis, saemix.plot.randeff, saemix.plot.obsvspre saemix.plot.fits, saemix.plot.parcov, saemix.plot.randeffcov, saemix.plot.distpsi, saemix.plot.scatterresiduals, saemix.plot.distribresiduals, saemix.plot.vpc

Examples

```
data(theo.saemix)

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")

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.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="constant")

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE)

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

saemix.plot.select(saemix.fit,data=TRUE,main="Spaghetti plot of data")

# Putting several graphs on the same plot
par(mfrow=c(2,2))
saemix.plot.select(saemix.fit,data=TRUE,vpc=TRUE,
  observations.vs.predictions=TRUE, new=FALSE)
```

---

saemix.plot.setoptions

Function setting the default options for the plots in SAEM

Description

This function can be used to create a list containing the default options and arguments used by the plot functions.

Usage

```
saemix.plot.setoptions(saemixObject)
```

Arguments

`saemixObject` an object returned by the `saemix` function

Details

A description of the options set via this list is provided in the PDF documentation.

ablinecol Color of the lines added to the plots (default: "DarkRed")

ablinelty Type of the lines added to the plots. Defaults to 2 (dashed line)

ablinelwd Width of the lines added to the plots (default: 2)

ask A logical value. If TRUE, users will be prompted before each new plot. Defaults to FALSE

cex A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default. Defaults to 1 (no magnification)

cex.axis Magnification to be used for axis annotation relative to the current setting of 'cex'. Defaults to 1 (no magnification)

cex.main Magnification to be used for main titles relative to the current setting of 'cex'. Defaults to 1 (no magnification)

cex.lab Magnification to be used for x and y labels relative to the current setting of 'cex'. Defaults to 1 (no magnification)

col.fillmed For the VPC plots: color filling the prediction interval for the median. Defaults to "pink"

col.fillpi For the VPC plots: color filling the prediction interval for the limits of the prediction interval. Defaults to "slategray1"

col.lmed For the VPC plots: color of the line showing the median of the simulated data. Defaults to "indianred4"

col.lobs For the VPC plots: color of the lines showing the median, 2.5 and 97.5th percentiles (for a 95

col.lpi For the VPC plots: color of the line showing the boundaries of the prediction intervals. Defaults to "slategray4"

col.obs For the VPC plots: color used to plot the observations. Defaults to "steelblue4"

cov.name Name of the covariate to be used in the plots. Defaults to the first covariate in the model

cov.value Value of the covariate to be used in the plots. Defaults to NA, indicating that the median value of the covariate (for continuous covariates) or the reference category (for categorical covariates) will be used

ilist List of indices of subjects to be included in the individual plots (defaults to all subjects)

indiv.par a string, giving the type of the individual estimates ("map"= conditional mode, "eap"=conditional mean). Defaults to conditional mode

lcol Main line color (default: black)

line.smooth Type of smoothing when a smoothed line is used in the plot ("m": mean value, "l": linear regression; "s": natural splines). Several options may be combined, for instance "ls" will add both a linear regression line and a line representing the fit of a natural spline. Defaults to "s"

lty Line type. Defaults to 1, corresponding to a straight line

lty.lmed For the VPC plots: type of the line showing the median of the simulated data. Defaults to 2 (dashed)

lty.obs For the VPC plots: type of the line showing the observed data. Defaults to 1

lty.lpi For the VPC plots: type of the line showing the boundaries of the simulated data. Defaults to 2 (dashed)

lwd Line width (default: 1)

lwd.lmed For the VPC plots: thickness of the line showing the median of the simulated data. Defaults to 2

lwd.obs For the VPC plots: thickness of the line showing the median and boundaries of the observed data. Defaults to 2

lwd.lpi For the VPC plots: thickness of the line showing the boundaries of the simulated data. Defaults to 1

par.name Name of the parameter to be used in the plots. Defaults to the first parameter in the model

pch Symbol type. Defaults to 20, corresponding to small dots

pcol Main symbol color (default: black)

range Range (expressed in number of SD) over which to plot the marginal distribution. Defaults to 4, so that the random effects for the marginal distribution is taken over the range [-4 SD; 4 SD]

res.plot Type of residual plot ("res.vs.x": scatterplot versus X, "res.vs.pred": scatterplot versus predictions, "hist": histogram, "qqplot": QQ-plot) (default: "res.vs.x")

smooth When TRUE, smoothed lines are added in the plots of predictions versus observations (default: FALSE)

tit Title of the graph (default: none)

type Type of the plot (as in the *R* plot function. Defaults to "b", so that both lines and symbols are shown

units Name of the predictor used in the plots (X). Defaults to the name of the first predictor in the model (saemix.data$names$predictors[1])

vpc.bin Number of binning intervals when plotting the VPC (the (vpc.bin-1) breakpoints are taken as the empirical quantiles of the X data). Defaults to 10

vpc.interval Size of the prediction intervals.Defaults to 0.95 for the 95% prediction interval

vpc.obs Should the observations be overlayed on the VPC plot. Defaults to TRUE

vpc.pi Should prediction bands be computed around the median and the bounds of the prediction intervals for the VPC. Defaults to TRUE

xlab Label for the X-axis. Defaults to the name of the X predictor followed by the unit in bracket (eg "Time (hr)")

xlim Range for the X-axis. Defaults to NA, indicating that the range is to be set by the plot function

xlog A logical value. If TRUE, a logarithmic scale is in use. Defaults to FALSE

xname Name of the predictor used in the plots (X)

ylab Label for the Y-axis. Defaults to the name of the response followed by the unit in bracket (eg "Concentration (mg/L)" (Default: none)

ylim Range for the Y-axis. Defaults to NA, indicating that the range is to be set by the plot function

ylog A logical value. If TRUE, a logarithmic scale is in use. Defaults to FALSE

Value

A list containing the options set at their default value. This list can be stored in an object and its elements modified to provide suitable graphs.

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

SaemixObject,saemix, saemix.plot.data, saemix.plot.convergence, saemix.plot.llis, saemix.plot.randeff, saemix.plot.obsvspred, saemix.plot.fits, saemix.plot.parcov, saemix.plot.distpsi, saemix.plot.scatterresiduals, saemix.plot.vpc

Examples

```
# Theophylline example, after a call to fit.saemix (see examples)
# Not run
# sopt<-saemix.plot.setoptions(saemix.fit)
# sopt$ask<-TRUE
```

---

saemix.plots *General plot function from SAEM*

Description

Several plots (selectable by the type argument) are currently available: convergence plot, individual plots, predictions versus observations, distribution plots, VPC, residual plots.

Usage

```
plot(x,y, ...)
```

Arguments

x	an object returned by the saemix function
y	empty
...	optional arguments passed to the plots

Details

This is the generic plot function for an SaemixObject object, which implements different graphs related to the algorithm (convergence plots, likelihood estimation) as well as diagnostic graphs. A description is provided in the PDF documentation. Arguments such as main, xlab, etc... that can be given to the generic plot function may be used, and will be interpreted according to the type of plot that is to be drawn.

A special argument plot.type can be set to determine the type of plot; it can be one of:

data: A spaghetti plot of the data, displaying the observed data y as a function of the regression variable (time for a PK application)

convergence: For each parameter in the model, this plot shows the evolution of the parameter estimate versus the iteration number

likelihood: Graph showing the evolution of the log-likelihood during the estimation by importance sampling

observations.vs.predictions: Plot of the predictions computed with the population parameters versus the observations (left), and plot of the predictions computed with the individual parameters versus the observations (right)

residuals.scatter: Scatterplot of the residuals versus the predictor (top) and versus predictions (bottom), for weighted residuals (population residuals, left), individual weighted residuals (middle) and npde (right).

residuals.distribution: Distribution of the residuals, plotted as histogram (top) and as a QQ-plot (bottom), for weighted residuals (population residuals, left), individual weighted residuals (middle) and npde (right).

individual.fit: Individual fits are obtained using the individual parameters with the individual covariates

population.fit: Population fits are obtained using the population parameters with the individual covariates

both.fit: Individual fits, superposing fits obtained using the population parameters with the individual covariates (red) and using the individual parameters with the individual covariates (green)

marginal.distribution: Distribution of the parameters (conditional on covariates when some are included in the model). A histogram of individual parameter estimates can be overlayed on the plot, but it should be noted that the histogram does not make sense when there are covariates influencing the parameters and a warning will be displayed

random.effects: Boxplot of the random effects

correlations: Correlation between the random effects

parameters.vs.covariates: Plots of the estimates of the individual parameters versus the covariates, using scatterplot for continuous covariates, boxplot for categorical covariates

randeff.vs.covariates: Plots of the estimates of the random effects versus the covariates, using scatterplot for continuous covariates, boxplot for categorical covariates

npde: Plots 4 graphs to evaluate the shape of the distribution of the normalised prediction distribution errors (npde)

vpc: Visual Predictive Check, with options to include the prediction intervals around the boundaries of the selected interval as well as around the median (50th percentile of the simulated data).

In addition, the following values for plot.type produce a series of plots:

reduced: produces the following plots: plot of the data, convergence plots, plot of the likelihood by importance sampling (if computed), plots of observations versus predictions. This is the default behaviour of the plot function applied to an SaemixObject object

full: produces the following plots: plot of the data, convergence plots, plot of the likelihood by importance sampling (if computed), plots of observations versus predictions, scatterplots and distribution of residuals, VPC, npde, boxplot of the random effects, distribution of the parameters, correlations between random effects, plots of the relationships between individually estimated parameters and covariates, plots of the relationships between individually estimated random effects and covariates

Each plot can be customised by modifying options, either through a list of options set by the `saemix.plot.setoptions` function, or on the fly by passing an option in the call to the plot (see examples).

Value

None

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

SaemixObject, saemix, saemix.plot.setoptions, saemix.plot.select, saemix.plot.data

Examples

```
data(theo.saemix)

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")

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.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="constant")

saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE)

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

# Set of default plots
plot(saemix.fit)

# Data
plot(saemix.fit,plot.type="data")

# Convergence
plot(saemix.fit,plot.type="convergence")

# Individual plot for subject 1, smoothed
plot(saemix.fit,plot.type="individual.fit",ilist=1,smooth=TRUE)

# Individual plot for subject 1 to 12, with ask set to TRUE
# (the system will pause before a new graph is produced)
plot(saemix.fit,plot.type="individual.fit",ilist=c(1:12),ask=TRUE)

# Diagnostic plot: observations versus population predictions
par(mfrow=c(1,1))
plot(saemix.fit,plot.type="observations.vs.predictions",level=0,new=FALSE)

# LL by Importance Sampling
```

```
plot(saemix.fit,plot.type="likelihood")

# Scatter plot of residuals
# Data will be simulated to compute weighted residuals and npde
# the results shall be silently added to the object saemix.fit
plot(saemix.fit,plot.type="residuals.scatter")

# Boxplot of random effects
plot(saemix.fit,plot.type="random.effects")

# Relationships between parameters and covariates
plot(saemix.fit,plot.type="parameters.vs.covariates")

# Relationships between parameters and covariates, on the same page
par(mfrow=c(3,2))
plot(saemix.fit,plot.type="parameters.vs.covariates",new=FALSE)

# VPC
plot(saemix.fit,plot.type="vpc")
```

`saemixControl` *List of options for running the algorithm SAEM*

Description

List containing the variables relative to the optimisation algorithm. All these elements are optional and will be set to default values when running the algorithm if they are not specified by the user.

Usage

```
saemixControl(algorithms = c(1, 1, 1), nbiter.saemix = c(300, 100),
  nb.chains = 1, fix.seed = TRUE, seed = 23456, nmc.is = 5000, nu.is = 4,
  print.is = FALSE, nbdisplay = 100, displayProgress = TRUE, nbiter.burn = 5,
  nbiter.mcmc = c(2, 2, 2), proba.mcmc = 0.4, stepsize.rw = 0.4, rw.init = 0.5,
  alpha.sa = 0.97, nnodes.gq = 12, nsd.gq = 4, maxim.maxiter = 100,
  nb.sim = 1000, nb.simpred = 100, ipar.lmcmc = 50, ipar.rmcmc = 0.05,
  print = TRUE, save = TRUE, save.graphs = TRUE, directory = "newdir",
  warnings = FALSE)
```

Arguments

A number of variables relative to the optimisation algorithm can be set before running the SAEM algorithm; all options not set by the user are given default values as indicated below:

a vector of 1s specifying which algorithms are to be run. Defaults to c(1,1,1) (respectively estimation of the Fisher Information Matrix (by linearisation), estimation of the individual parameters, and estimation of the log-likelihood by importance sampling)

`nbiter.saemix`

`algorithms`	nb of iterations in each step (a vector containing 2 elements)
`nb.chains`	nb of chains to be run in parallel in the MCMC algorithm. Defaults to 1.
`nbiter.burn`	nb of iterations for burning

`nbiter.mcmc` nb of iterations in each kernel during the MCMC step

`proba.mcmc` probability of acceptance

`stepsize.rw` stepsize for kernels q2 and q3. Defaults to 0.4

`rw.init` initial variance parameters for kernels. Defaults to 0.5

`alpha.sa` parameter controlling cooling in the Simulated Annealing algorithm. Defaults to 0.97

`fix.seed` TRUE (default) to use a fixed seed for the random number generator. When FALSE, the random number generator is initialised using a new seed, created from the current time. Hence, different sessions started at (sufficiently) different times will give different simulation results. The seed is stored in the element seed of the options list.

`seed` seed for the random number generator. Defaults to 123456

`nmc.is` nb of samples used when computing the likelihood through importance sampling

`nu.is` number of degrees of freedom of the Student distribution used for the estimation of the log-likelihood by Importance Sampling. Defaults to 4

`print.is` when TRUE, a plot of the likelihood as a function of the number of MCMC samples when computing the likelihood through importance sampling is produced and updated every 500 samples. Defaults to FALSE

`nbdisplay` nb of iterations after which to display progress

`displayProgress` when TRUE, the convergence plots are plotted after every nbdisplay iteration, and a dot is written in the terminal window to indicate progress. When FALSE, plots are not shown and the algorithm runs silently. Defaults to TRUE

`nnodes.gq` number of nodes to use for the Gaussian quadrature when computing the likelihood with this method (defaults to 12)

`nsd.gq` span (in SD) over which to integrate when computing the likelihood by Gaussian quadrature. Defaults to 4 (eg 4 times the SD)

`maxim.maxiter` Maximum number of iterations to use when maximising the fixed effects in the algorithm. Defaults to 100

`nb.sim` number of simulations to perform to produce the VPC plots or compute npde. Defaults to 1000

`nb.simpred` number of simulations used to compute mean predictions (ypred element), taken as a random sample within the nb.sim simulations used for npde

`ipar.lmcmc` number of iterations required to assume convergence for the conditional estimates. Defaults to 50

`ipar.rmcmc` confidence interval for the conditional mean and variance. Defaults to 0.95

`print` whether the results of the fit should be printed out. Defaults to TRUE

`save` whether the results of the fit should be saved to a file. Defaults to TRUE

`save.graphs` whether diagnostic graphs and individual graphs should be saved to files. Defaults to TRUE

`directory` the directory in which to save the results. Defaults to "newdir" in the current directory

`warnings` whether warnings should be output during the fit. Defaults to FALSE

Details

All the variables are optional and will be set to their default value when running saemix.

The function saemix returns an object with an element options containing the options used for the algorithm, with defaults set for elements which have not been specified by the user.

These elements are used in subsequent functions and are not meant to be used directly.

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

SaemixData,SaemixModel, SaemixObject, saemix

Examples

```
# All default options
saemix.options<-saemixControl()

# All default options, changing seed
saemix.options<-saemixControl(seed=632545)
```

---

saemixData *Function to create a SaemixData object*

Description

This function creates a SaemixData object. The only mandatory argument is the name of the dataset. If the dataset has a header (or named columns), the program will attempt to detect which column correspond to ID, predictor(s) and response. Warning messages will be printed during the object creation and should be read for details.

Usage

```
saemixData(name.data, header, sep, na, name.group, name.predictors,
  name.response, name.X, name.covariates = c(),
  units = list(x = "", y = "", covariates = c()))
```

Arguments

name.data	name of the dataset (can be a character string giving the name of a file on disk or of a dataset in the R session, or the name of a dataset
header	whether the dataset/file contains a header. Defaults to TRUE
sep	the field separator character. Defaults to any number of blank spaces ("")
na	a character vector of the strings which are to be interpreted as NA values. Defaults to c(NA)
name.group	name (or number) of the column containing the subject id
name.predictors	name (or number) of the column(s) containing the predictors (the algorithm requires at least one predictor x)
name.response	name (or number) of the column containing the response variable y modelled by predictor(s) x
name.covariates	name (or number) of the column(s) containing the covariates, if present (otherwise missing)
name.X	name of the column containing the regression variable to be used on the X axis in the plots (defaults to the first predictor)
units	list with up to three elements, x, y and optionally covariates, containing the units for the X and Y variables respectively, as well as the units for the different covariates (defaults to empty)

Details

This function is the user-friendly constructor for the SaemixData object class.

Value

A SaemixData object (see saemixData).

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

SaemixData,SaemixModel, saemixControl,saemix

Examples

```
data(theo.saemix)

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")

print(saemix.data)

plot(saemix.data)
```

SaemixData-class *Class "SaemixData"*

Description

An object of the SaemixData class, representing a longitudinal data structure, used by the SAEM algorithm.

Objects from the Class

An object of the SaemixData class can be created by using the function `saemixData`

Slots

`name.data`: Object of class `"character"`: name of the dataset

`header`: Object of class `"logical"`: whether the dataset/file contains a header. Defaults to TRUE

`sep`: Object of class `"character"`: the field separator character

`na`: Object of class `"character"`: a character vector of the strings which are to be interpreted as NA values

`name.group`: Object of class `"character"`: name of the column containing the subject id

`name.predictors`: Object of class `"character"`: name of the column(s) containing the predictors

`name.response`: Object of class `"character"`: name of the column containing the response variable y modelled by predictor(s) x

`name.covariates`: Object of class `"character"`: name of the column(s) containing the covariates, if present (otherwise empty)

`name.X`: Object of class `"character"`: name of the column containing the regression variable to be used on the X axis in the plots

`units`: Object of class `"list"`: list with up to three elements, x, y and optionally covariates, containing the units for the X and Y variables respectively, as well as the units for the different covariates

`tab`: Object of class `"data.frame"`: dataframe containing the data

`N`: Object of class `"numeric"`: number of subjects

`index`: Object of class `"numeric"`: subject index

`id`: Object of class `"character"`: subject id data

`xind`: Object of class `"data.frame"`: predictors data (eg dose, time)

`y`: Object of class `"numeric"`: response data

`yorig`: Object of class `"numeric"`: response data, on the original scale

`cov`: Object of class `"data.frame"`: covariates, if present in the model (otherwise empty)

`ntot.obs`: Object of class `"numeric"`: total number of observations

`nind.obs`: Object of class `"numeric"`: vector containing the number of observations for each subject

Methods

[<- `signature(x = "SaemixData")`: replace elements of object

[`signature(x = "SaemixData")`: access elements of object

initialize `signature(.Object = "SaemixData")`: internal function to initialise object, not to be used

plot `signature(x = "SaemixData")`: plot the data

print `signature(x = "SaemixData")`: details about the object

read.saemixData `signature(object = "SaemixData")`: internal function, not to be used

showall `signature(object = "SaemixData")`: all the elements in the object

show `signature(object = "SaemixData")`: details about the object

summary `signature(object = "SaemixData")`: summary of the data

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

saemixData,SaemixModel, saemixControl,saemix

Examples

```
showClass("SaemixData")

# Specifying column names
data(theo.saemix)
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")

# Specifying column numbers
data(theo.saemix)
saemix.data<-saemixData(name.data=theo.saemix,header=TRUE,sep=" ",na=NA,
```

```
  name.group=1,name.predictors=c(2,3),name.response=c(4), name.covariates=5:6,
  units=list(x="hr",y="mg/L",covariates=c("kg","-")), name.X="Time")

# No column names specified, using automatic recognition of column names
data(PD1.saemix)
saemix.data<-saemixData(name.data=PD1.saemix,header=TRUE,
  name.covariates=c("gender"),units=list(x="mg",y="-",covariates=c("-")))
```

`saemixModel` *Function to create a SaemixModel object*

Description

This function creates a SaemixModel object. The two mandatory arguments are the name of a R function computing the model in the SAEMIX format (see details and examples) and a matrix psi0 giving the initial estimates of the fixed parameters in the model, with one row for the population mean parameters and one row for the covariate effects (see documentation).

Usage

```
saemixModel(model, psi0, description = "", error.model = character(),
  transform.par = numeric(), fixed.estim = numeric(),
  covariate.model = matrix(nrow = 0, ncol = 0),
  covariance.model = matrix(nrow = 0, ncol = 0),
  omega.init = matrix(nrow = 0, ncol = 0), error.init = numeric(),
  name.modpar = character())
```

Arguments

`model`	name of the function used to compute the structural model. The function should return a vector of predicted values given a matrix of individual parameters, a vector of indices specifying which records belong to a given individual, and a matrix of dependent variables (see example below).
`psi0`	a matrix with a number of columns equal to the number of parameters in the model, and one (when no covariates are available) or two (when covariates enter the model) giving the initial estimates for the fixed effects. The column names of the matrix should be the names of the parameters in the model, and will be used in the plots and the summaries. When only the estimates of the mean parameters are given, psi0 may be a named vector.
`description`	a character string, giving a brief description of the model or the analysis
`error.model`	type of residual error model (valid types are constant, proportional, combined and exponential). Defaults to constant
`transform.par`	the distribution for each parameter (0=normal, 1=log-normal, 2=probit, 3=logit). Defaults to a vector of 1s (all parameters have a log-normal distribution)
`fixed.estim`	whether parameters should be estimated (1) or fixed to their initial estimate (0). Defaults to a vector of 1s
`covariate.model`	a matrix giving the covariate model. Defaults to no covariate in the model

`covariance.model`
: a square matrix of size equal to the number of parameters in the model, giving the variance-covariance matrix of the model: 1s correspond to estimated variances (in the diagonal) or covariances (off-diagonal elements). Defaults to the identity matrix

`omega.init`
: a square matrix of size equal to the number of parameters in the model, giving the initial estimate for the variance-covariance matrix of the model. Defaults to the identity matrix

`error.init`
: a vector of size 2 giving the initial value of a and b in the error model. Defaults to 1 for each estimated parameter in the error model

`name.modpar`
: names of the model parameters, if they are not given as the column names (or names) of psi0

Details

This function is the user-friendly constructor for the SaemixModel object class.

Value

A SaemixModel object (see `saemixModel`).

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

`SaemixData`,`SaemixModel`, `saemixControl`,`saemix`

Examples

```
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.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="constant")
```

SaemixModel-class *Class "SaemixModel"*

Description

An object of the SaemixModel class, representing a non-linear mixed-effect model structure, used by the SAEM algorithm.

Objects from the Class

An object of the SaemixModel class can be created by using the function saemixModel

Slots

model: function used to compute the structural model. The function should return a vector of predicted values given a matrix of individual parameters, a vector of indices specifying which records belong to a given individual, and a matrix of dependent variables (see examples).

description: model description (optional) as a character string

psi0: a matrix with a number of columns equal to the number of parameters in the model, and one (when no covariates are available) or two (when covariates enter the model) giving the initial estimates for the fixed effects. The column names of the matrix should be the names of the parameters in the model, and will be used in the plots and the summaries

transform.par: the distribution for each parameter (0=normal, 1=log-normal, 2=probit, 3=logit). Defaults to a vector of 1s (all parameters have a log-normal distribution)

fixed.estim: whether parameters should be estimated (1) or fixed to their initial estimate (0). Defaults to a vector of 1s

error.model: name of the residual error model

covariate.model: a matrix giving the covariate model. Defaults to no covariate in the model (empty matrix)

betaest.model: a matrix giving the effects model (internal)

covariance.model: a square matrix of size equal to the number of parameters in the model, giving the variance-covariance matrix of the model: 1s correspond to estimated variances (in the diagonal) or covariances (off-diagonal elements). Defaults to the identity matrix

omega.init: a square matrix of size equal to the number of parameters in the model, giving the initial estimate for the variance-covariance matrix of the model. Defaults to the identity matrix

error.init: a vector of size 2 giving the initial value of a and b in the error model. Defaults to 1 for each estimated parameter in the error model

nb.parameters: number of parameters

name.modpar: names of the model parameters

name.fixed: names of the fixed effects estimated in the model

name.random: names of the random effects estimated in the model

name.res: names of the parameters of the residual error model

name.predictors: names of the predictors (X)

name.X: name of the predictor used in graphs

name.response: name of the response (Y)

`name.cov`: name of the covariates

`indx.fix`: index of estimated fixed effects (internal)

`indx.cov`: index of estimated fixed effects associated with covariate effects (internal)

`indx.omega`: index of estimated random effects (internal)

`indx.res`: index of parameters of the residual error model (internal)

`Mcovariates`: matrix of the covariates (internal)

Methods

[<- `signature(x = "SaemixModel")`: replace elements of object

[`signature(x = "SaemixModel")`: access elements of object

initialize `signature(.Object = "SaemixModel")`: internal function to initialise object, not to be used

plot `signature(x = "SaemixModel")`: plot results (see `saemix.plot.data`

print `signature(x = "SaemixModel")`: prints details about the object

showall `signature(object = "SaemixModel")`: prints an extensive summary of the object

show `signature(object = "SaemixModel")`: prints a short summary of the object

summary `signature(object = "SaemixModel")`: summary of the model

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (`http://software.monolix.org/sdoms/software/`)

See Also

`SaemixData`,`saemixModel`, `saemixControl`,`saemix`

Examples

```
showClass("SaemixModel")
```

SaemixObject-class *Class "SaemixObject"*

Description

An object of the SaemixObject class, storing the results obtained by a call to the SAEM algorithm

Details

Details of the algorithm can be found in the pdf file accompanying the package.

Objects from the Class

Objects are created by a call `saemix()`.

Slots

`data`: Object of class `"SaemixData"` an object of the SaemixData class, representing a longitudinal data structure, used by the SAEM algorithm. See `SaemixData`

`model`: Object of class `"SaemixModel"` an object of the SaemixModel class, representing the structure a non-linear mixed effect model, used by the SAEM algorithm. See `SaemixModel`

`results`: Object of class `"SaemixRes"` ~~

`rep.data`: Object of class `"SaemixRepData"` (internal use only) the data part, replicated a number of times equal to the number of chains used in the SAEM algorithm (see documentation for details). Not intended to be accessed to directly by the user.

`sim.data`: Object of class `"SaemixSimData"` (internal use only) data simulated according to the design in data, with the model in model and the parameters estimated by the SAEM algorithm, after a call to simul.saemix (see documentation for details). Not intended to be accessed to directly by the user.

`options`: Object of class `"list"` a list of options containing variables controlling the algorithm

`prefs`: Object of class `"list"` a list of graphical preferences applied to plots

Methods

[<- `signature(x = "SaemixObject")`: replace elements of object

[`signature(x = "SaemixObject")`: access elements of object

initialize `signature(.Object = "SaemixObject")`: internal function to initialise object, not to be used

plot `signature(x = "SaemixObject")`: plot results (see `saemix.plot.data`

predict `signature(object = "SaemixObject")`: compute model predictions

print `signature(x = "SaemixObject")`: prints details about the object

showall `signature(object = "SaemixObject")`: prints an extensive summary of the object

show `signature(object = "SaemixObject")`: prints a short summary of the object

summary `signature(object = "SaemixObject")`: summary of the results

coef `signature(object = "SaemixObject")`: extracts coefficients. Returns a list with components fixed (estimated fixed effects), population (population parameter estimates, including covariate effects: a list with two components map and cond), individual (individual parameter estimates: a list with two components map and cond). For population and individual, the map component of the list gives the MAP estimates (the mode of the distribution) while the cond component gives the conditional mean estimates. Some components may be missing (eg MAP estimates) if they have not been computed during or after the fit.

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. Computational Statistics and Data Analysis 49, 4 (2005), 1020-1038.

Monolix32_UsersGuide.pdf (http://software.monolix.org/sdoms/software/)

See Also

SaemixData,SaemixModel, saemixControl,saemix,plot.saemix, saemix.plot.data

Examples

```
showClass("SaemixObject")
```

---

show-methods *Methods for Function show*

Description

Prints a short summary of an object

Methods

`signature(x = "ANY")` Default show function

`signature(x = "SaemixData")` Prints a short summary of a SaemixData object

`signature(x = "SaemixModel")` Prints a short summary of a SaemixModel object

`signature(x = "SaemixObject")` Prints a short summary of the results from a SAEMIX fit

`signature(x = "SaemixRes")` Not user-level

`signature(object = "SaemixRepData")` Prints a short summary of a SaemixRepData object

`signature(object = "SaemixSimData")` Prints a short summary of a SaemixSimData object

showall *Prints out an extensive summary of an object*

Description

This function shows an extensive summary of an object, and is used mainly to visualise the majority of the elements of an object

Usage

```
showall(object)
```

Arguments

object	showall methods are available for objects of typeSaemixData, SaemixModel and SaemixObject

Details

None

Value

None

See Also

SaemixData,SaemixModel, SaemixObject

Examples

```
# A SaemixData object
data(theo.saemix)
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")
showall(saemix.data)

# A 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)
}
saemix.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
```

```
  covariate.model=matrix(c(0,1,0,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="constant")
showall(saemix.model)
```

---

showall-methods *Methods for Function showall*

Description

Prints an extensive summary of an object

Methods

`signature(x = "SaemixData")` Prints a extensive summary of a SaemixData object

`signature(x = "SaemixModel")` Prints a extensive summary of a SaemixModel object

`signature(x = "SaemixObject")` Prints a extensive summary of the results from a SAEMIX fit

`signature(x = "SaemixRes")` Not user-level

---

simul.saemix *Perform simulations under the model*

Description

This function is used to simulate from the model. It can be called with the estimated parameters (the default), the initial parameters, or with a set of parameters. The original design can be used in the simulations, or a different dataset may be used with the same structure (covariates) as the original design. This function is not yet implemented.

Usage

```
simul.saemix(saemixObject, nsim = saemixObject["options"]$nb.sim,
  predictions = TRUE, res.var = TRUE, uncertainty = FALSE)
```

Arguments

`saemixObject`	an object returned by the `saemix` function
`nsim`	Number of simulations to perform. Defaults to the nb.simpred element in options
`predictions`	Whether the simulated parameters should be used to compute predictions. Defaults to TRUE
`res.var`	Whether residual variability should be added to the predictions. Defaults to TRUE
`uncertainty`	Uses uncertainty (currently not implemented). Defaults to FALSE

Details

This function is used to produce Visual Predictive Check graphs, as well as to compute the normalised prediction distribution errors (npde).

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>, Audrey Lavenu, Marc Lavielle.

References

Brendel, K, Comets, E, Laffont, C, Laveille, C, Mentre, F. Metrics for external model evaluation with an application to the population pharmacokinetics of gliclazide, Pharmaceutical Research 23 (2006), 2036-2049.

Holford, N. The Visual Predictive Check: superiority to standard diagnostic (Rorschach) plots (Abstract 738), in: 14th Meeting of the Population Approach Group in Europe, Pamplona, Spain, 2005.

See Also

SaemixObject, saemix, saemix.plot.data, saemix.plot.convergence, saemix.plot.llis, saemix.plot.randeff, saemix.plot.obsvspred, saemix.plot.fits, saemix.plot.parcov, saemix.plot.distpsi, saemix.plot.scatterresiduals, saemix.plot.vpc

---

summary-methods *Methods for Function summary*

Description

Methods for function summary

Methods

signature(x = "ANY") default summary function ?

signature(x = "SaemixData") summary of the data

signature(x = "SaemixModel") summary of the model

signature(x = "SaemixObject") summary of an SaemixObject

---

testnpde *Tests for normalised prediction distribution errors*

Description

Performs tests for the normalised prediction distribution errors returned by npde

Usage

```
testnpde(npde)
```

Arguments

npde	the vector of prediction distribution errors

Details

Given a vector of normalised prediction distribution errors (npde), this function compares the npde to the standardised normal distribution N(0,1) using a Wilcoxon test of the mean, a Fisher test of the variance, and a Shapiro-Wilks test for normality. A global test is also reported.

The helper functions kurtosis and skewness are called to compute the kurtosis and skewness of the distribution of the npde.

Value

a list containing 4 components:

Wilcoxon test of mean=0	compares the mean of the npde to 0 using a Wilcoxon test
variance test	compares the variance of the npde to 1 using a Fisher test
SW test of normality	compares the npde to the normal distribution using a Shapiro-Wilks test
global test	an adjusted p-value corresponding to the minimum of the 3 previous p-values multiplied by the number of tests (3), or 1 if this p-value is larger than 1.

Author(s)

Emmanuelle Comets <emmanuelle.comets@inserm.fr>

References

K. Brendel, E. Comets, C. Laffont, C. Laveille, and F. Mentr\'e. Metrics for external model evaluation with an application to the population pharmacokinetics of gliclazide. *Pharmaceutical Research*, 23:2036–49, 2006.

See Also

saemix, saemix.plot.npde

---

theo.saemix *Pharmacokinetics of theophylline, in SAEM format*

Description

The theo.saemix data frame has 132 rows and 6 columns of data from an experiment on the pharmacokinetics of theophylline. A column with gender was added to the original data for demo purposes, and contains simulated data.

Usage

```
theo.saemix
```

Format

This data frame contains the following columns:

Id: an ordered factor with levels `1`, ..., `12` identifying the subject on whom the observation was made. The ordering is by Time at which the observation was made.

Dose: dose of theophylline administered orally to the subject (mg/kg).

Time: time since drug administration when the sample was drawn (hr).

Concentration: theophylline concentration in the sample (mg/L).

Weight: weight of the subject (kg).

Sex: gender of the subject (0=men, 1=women).

Details

Boeckmann, Sheiner and Beal (1994) report data from a study by Dr. Robert Upton of the kinetics of the anti-asthmatic drug theophylline. Twelve subjects were given oral doses of theophylline then serum concentrations were measured at 11 time points over the next 25 hours. In the present package *npde*, we removed the data at time 0.

These data are analyzed in Davidian and Giltinan (1995) and Pinheiro and Bates (2000) using a two-compartment open pharmacokinetic model.

These data are also available in the library `datasets` under the name `Theoph` in a slightly modified format and including the data at time 0.

Source

Boeckmann, A. J., Sheiner, L. B. and Beal, S. L. (1994), *NONMEM Users Guide: Part V*, NONMEM Project Group, University of California, San Francisco.

Davidian, M. and Giltinan, D. M. (1995) *Nonlinear Models for Repeated Measurement Data*, Chapman & Hall (section 5.5, p. 145 and section 6.6, p. 176)

Pinheiro, J. C. and Bates, D. M. (2000) *Mixed-effects Models in S and S-PLUS*, Springer (Appendix A.29)

Examples

```
data(theo.saemix)
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")

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)
}
# Default model, no covariate
saemix.model<-saemixModel(model=model1cpt,
```

```
  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"))),transform.par=c(1,1,1))

# Note: remove the options save=FALSE and save.graphs=FALSE
# to save the results and graphs
saemix.options<-list(seed=632545,save=FALSE,save.graphs=FALSE)

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

# Model with covariates
saemix.model<-saemixModel(model=model1cpt,
  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"))),transform.par=c(1,1,1),
  covariate.model=matrix(c(0,0,1,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
  covariance.model=matrix(c(1,0,0,0,1,1,0,1,1),ncol=3,byrow=TRUE),
  omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="combined")

saemix.options<-list(seed=39546,save=FALSE,save.graphs=FALSE)

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

`yield.saemix` *Wheat yield in crops treated with fertiliser, in SAEM format*

Description

`yield.saemix` contains data from winter wheat experiments.

Usage

```
yield.saemix
```

Format

This data frame contains the following columns:

site: the site number

dose: dose of nitrogen fertiliser (kg/ha).

yield: grain yield (kg/ha).

soil.nitrogen: end-of-winter mineral soil nitrogen (NO3- plus NH4+) in the 0 to 90 cm layer was measured on each site/year (kg/ha).

Details

The data in the `yield.saemix` comes from 37 winter wheat experiments carried out between 1990 and 1996 on commercial farms near Paris, France. Each experiment was from a different site. Two soil types were represented, a loam soil and a chalky soil. Common winter wheat varieties were used. Each experiment consisted of five to eight different nitrogen fertiliser rates, for a total of 224 nitrogen treatments. Nitrogen fertilizer was applied in two applications during the growing season. For each nitrogen treatment, grain yield (adjusted to 150 g.kg-1 grain moisture content) was measured. In addition, end-of-winter mineral soil nitrogen (NO3- plus NH4+) in the 0 to 90

cm layer was measured on each site-year during February when the crops were tillering. Yield and end-of-winter mineral soil nitrogen measurements were in the ranges 3.44-11.54 t.ha-1 , and 40-180 kg.ha-1 respectively.

Source

Makowski, D., Wallach, D., and Meynard, J.-M (1999). Models of yield, grain protein, and residual mineral nitrogen responses to applied nitrogen for winter wheat. Agronomy Journal 91: 377-385.

Examples

```
data(yield.saemix)
saemix.data<-saemixData(name.data=yield.saemix,header=TRUE,name.group=c("site"),
  name.predictors=c("dose"),name.response=c("yield"),
  name.covariates=c("soil.nitrogen"),units=list(x="kg/ha",y="t/ha",
  covariates=c("kg/ha")))

# Model: linear + plateau
yield.LP<-function(psi,id,xidep) {
# input:
#   psi : matrix of parameters (3 columns, ymax, xmax, slope)
#   id : vector of indices
#   xidep : dependent variables (same nb of rows as length of id)
# returns:
#   a vector of predictions of length equal to length of id
  x<-xidep[,1]
  ymax<-psi[id,1]
  xmax<-psi[id,2]
  slope<-psi[id,3]
  f<-ymax+slope*(x-xmax)
#  cat(length(f),"  ",length(ymax),"\n")
  f[x>xmax]<-ymax[x>xmax]
  return(f)
}
saemix.model<-saemixModel(model=yield.LP,description="Linear plus plateau model",
  psi0=matrix(c(8,100,0.2,0,0,0),ncol=3,byrow=TRUE,dimnames=list(NULL,
  c("Ymax","Xmax","slope"))),covariate.model=matrix(c(0,0,0),ncol=3,byrow=TRUE),
  transform.par=c(0,0,0),covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,
  byrow=TRUE),error.model="constant")

saemix.options<-list(algorithms=c(1,1,1),nb.chains=1,seed=666,
  save=FALSE,save.graphs=FALSE)

# Plotting the data
plot(saemix.data,xlab="Fertiliser dose (kg/ha)", ylab="Wheat yield (t/ha)")

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

# Plotting the data
plot(saemix.data,xlab="Fertiliser dose (kg/ha)", ylab="Wheat yield (t/ha)")

# Comparing the likelihoods obtained by linearisation and importance sampling
# to the likelihood obtained by Gaussian Quadrature
saemix.fit<-llgq.saemix(saemix.fit)
{
cat("LL by Importance sampling, LL_IS=",saemix.fit["results"]["ll.is"],"\n")
```

```
cat("LL by linearisation, LL_lin=",saemix.fit["results"]["ll.lin"],"\n")
cat("LL by Gaussian Quadrature, LL_GQ=",saemix.fit["results"]["ll.gq"],"\n")
}

# Testing for an effect of covariate soil.nitrogen on Xmax
saemix.model2<-saemixModel(model=yield.LP,description="Linear plus plateau model",
  psi0=matrix(c(8,100,0.2,0,0,0),ncol=3,byrow=TRUE,dimnames=list(NULL,
  c("Ymax","Xmax","slope"))),covariate.model=matrix(c(0,1,0),ncol=3,byrow=TRUE),
  transform.par=c(0,0,0),covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,
  byrow=TRUE),error.model="constant")

saemix.fit2<-saemix(saemix.model2,saemix.data,saemix.options)

# BIC for the two models
{
  cat("Model without covariate, BIC=",saemix.fit["results"]["bic.is"],"\n")
  cat("Model with covariate, BIC=",saemix.fit2["results"]["bic.is"],"\n")
  pval<-1-pchisq(-2*saemix.fit["results"]["ll.is"]+2*saemix.fit2["results"]["ll.is"],1)
  cat("        LRT: p=",pval,"\n")
}
```

[-methods *Methods for "get" Function [*

Description

Methods for function `[`, to allow access to the value of an element within an S4 object

Methods

`signature(x = "SaemixData")` access elements of an object SaemixData

`signature(x = "SaemixModel")` access elements of an object SaemixModel

`signature(x = "SaemixObject")` access elements of an object SaemixObject

`signature(x = "SaemixRepData")` access elements of an object SaemixRepData

`signature(x = "SaemixRes")` access elements of an object SaemixRes

`signature(x = "SaemixSimData")` access elements of an object SaemixSimData

[<-methods *Methods for "set" Function [<-*

Description

Methods for function `[<-`, to allow replacements of the value of an element within an S4 object

Methods

`signature(x = "SaemixData")` replace elements of an object SaemixData

`signature(x = "SaemixModel")` replace elements of an object SaemixModel

`signature(x = "SaemixObject")` replace elements of an object SaemixObject

`signature(x = "SaemixRepData")` replace elements of an object SaemixRepData

`signature(x = "SaemixRes")` replace elements of an object SaemixRes

`signature(x = "SaemixSimData")` replace elements of an object SaemixSimData

Index