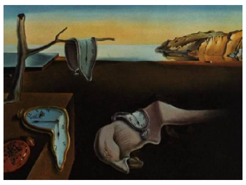


Journal of Statistical Software

August 2017, Volume 80, Issue 3. doi: 10.18637/jss.v080.i03

Parameter Estimation in Nonlinear Mixed Effect Models Using saemix, an R Implementation of the SAEM Algorithm

Emmanuelle Comets
INSERM CIC 1414
U. Rennes-I
INSERM, IAME, UMR 1137
U. Paris Diderot

Audrey Lavenu
U. Rennes-I
INSERM CIC 1414

Marc Lavielle
INRIA Saclay, Popix
U. Paris Sud

Abstract

The **saemix** package for R provides maximum likelihood estimates of parameters in nonlinear mixed effect models, using a modern and efficient estimation algorithm, the stochastic approximation expectation maximisation (SAEM) algorithm. In the present paper we describe the main features of the package, and apply it to several examples to illustrate its use. Making use of S4 classes and methods to provide user-friendly interaction, this package provides a new estimation tool to the R community.

Keywords: nonlinear mixed effect models, stochastic approximation EM algorithm, pharmacokinetics, pharmacodynamics, theophylline, orange tree, S4 classes.

1. Introduction

Over the recent years, a wealth of complex data has been accrued in many fields of biology, medicine or agriculture. These data often present a hierarchical structure, with correlations introduced by repeated measurements on the same individual. These correlations can be handled by modeling the evolution of a process with time and assuming subject-specific parameters to account for interindividual differences. The models used to describe the dynamics of biological processes are often nonlinear with respect to the parameters involved, and the statistical tools of choice in this context are nonlinear mixed effect models (Ette and Williams 2007). Estimating the parameters of the models can be done through maximum likelihood or Bayesian approaches; in the present paper we will focus on the first method. In maximum likelihood approaches, the estimation process consists in defining a statistical criterion, generally minimizing the likelihood or an approximation of the likelihood, and selecting an algorithm to obtain the minimum of this criterion.

Longitudinal data arise in many fields, such as agronomy, spatial analysis, imagery, clinical trials, and have been particularly prominent in the field of pharmacokinetics (PK) and pharmacodynamics (PD). As such, they play an important role in the process of developing new drugs, and PK/PD analysis is an integral part of the registration file submitted to the health authority for the approval of new drugs (Lee *et al.* 2011). It is also increasingly used to tailor drug treatment and guide dose adaptation in special populations such as renally impaired patients or children. Because of the importance of these models in PK/PD applications, the first estimation methods have been developed in pharmacometrics. In the context of maximum likelihood, a dedicated software, called **NONMEM**, was developed in the 70's, which handles specific characteristics of pharmacologic data such as dosage regimen and other variables measured during treatment (Boeckmann, Sheiner, and Beal 1994). The first algorithms implemented in this software relied on model linearization to obtain an approximation of the likelihood, which cannot be computed easily in nonlinear mixed effect models. This approximation is then maximized through iterative Newton-Raphson minimization, a general purpose algorithm involving the gradient of the function to be optimized. Different approximations to the likelihood have been proposed, including the first-order conditional method where the linearization takes place around the current individual estimates at each iteration (Lindstrom and Bates 1990). These estimation methods have also been implemented in mainstream statistical software such as SAS (`PROC NLMIXED`; SAS Institute Inc. 2013) and R (R Core Team 2017), where the **nlme** (Pinheiro and Bates 1995) package is now part of the base installation.

Linearization-based methods however have both statistical and practical shortcomings. First and foremost, they do not in fact converge to the maximum likelihood estimates. While bias is generally minor for fixed effects, variance components may be significantly biased, especially with large interindividual variability. This has been shown to increase the type I error of likelihood tests (Comets and Mentré 2001; Bertrand, Comets, and Mentré 2008), with the potential of building wrong models. Second, these methods suffer from severe bias when applied to non-continuous data, as shown by Molenberghs and Verbeke (2005), while stochastic algorithms have been shown to provide unbiased estimates (Savic, Mentré, and Lavielle 2011). In practice, linearization-based algorithms also exhibit convergence issues and can be tricky to use with complex models (Plan, Maloney, Mentré, Karlsson, and Bertrand 2012). Over the past decade, new and powerful estimation algorithms have therefore been proposed to estimate the parameters of nonlinear mixed effect models (Lavielle 2014).

The alternative to model linearization is to compute the likelihood through numerical or statistical approximations, which preserve the statistical properties of maximum likelihood estimators. An example is the Laplace approximation, which is equivalent to a one-point adaptive Gaussian quadrature, and has been implemented in **NONMEM**. In R, the **lme4** package uses this approximation (Bates, Mächler, Bolker, and Walker 2015) with a penalized least square approach for the estimation algorithm. A powerful alternative to gradient-based minimization algorithms is the EM algorithm, also an iterative algorithm, which has been developed in the context of missing data (Dempster, Laird, and Rubin 1977). The stochastic approximation expectation maximization (SAEM) algorithm, combining a stochastic approximation to the likelihood with an EM algorithm, has proven very efficient for nonlinear mixed effect models, quickly converging to the maximum likelihood estimators (Delyon, Lavielle, and Moulines 1999) and performing better than linearization-based algorithms (Girard and Mentré 2005). It has been implemented in the **Monolix** software (Lavielle 2014), which has enjoyed increasingly widespread use over the last few years, more recently in the **Statistics toolbox** of MATLAB (`nlmefitsa.m`; The MathWorks Inc. 2014), and which is also available in **NONMEM** version 7. In the present paper, we provide an implementation of the SAEM algorithm in the R software through the **saemix** package (Comets, Lavenu, and Lavielle 2017) available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=saemix`.

The package uses the S4 class system of R to provide a user-friendly input and output system, with methods like `summary` or `plot` for fitted objects. The package provides summaries of the results, individual parameter estimates, standard errors (obtained using a linearized computation of the Fisher information matrix) Wald tests for fixed effects, and a number of diagnostic plots, including VPC plots and npde (Brendel, Comets, Laffont, Laveille, and Mentré 2006). The log-likelihood can be computed by three methods: a linearization of the model, an importance sampling procedure, or a Gaussian quadrature. The diagnostic graphs can be tailored to the user's individual preferences by setting a number of options, and are easily exported to a file. In the present paper we first present the statistical models, then we describe the features of the package by applying it to several examples. We conclude by simulation studies assessing the performance of **saemix** and its operating characteristics.

2. Nonlinear mixed effect models

2.1. Statistical models

A typical longitudinal dataset consists of observations collected on N individuals (for instance, subjects in a clinical trial). We assume that n_i observations $\mathbf{y}_i = \{y_{i1}, \dots, y_{in_i}\}$ have been collected on subject i, at design points $\mathbf{x}_i = \{x_{i1}, \dots, x_{in_i}\}$ (in a clinical trial, x will typically be time or doses). The statistical model for observation y_{ij} is the following:

$$y_{ij} = f(\psi_i, x_{ij}) + g(\psi_i, \sigma, x_{ij})\epsilon_{ij}. \tag{1}$$

In Equation 1, the function f represents the structural model, describing the evolution of the process being modeled, while the function g characterizes the residual error model, more specifically its variance, and may depend on additional parameters σ. We assume that the variability between subjects is captured by the vector of individual parameters ψ_i, and that the errors ϵ_{ij} are random samples from the standard normal distribution, and are independent of the ψ_i.

When f is nonlinear with respect to the parameters ψ, Equation 1 describes the general form of nonlinear mixed effect models. In **saemix**, we make some additional assumptions. We assume that the individual parameters ψ_i can be modeled parametrically as a function of fixed effects μ, individual random effects η_i, and subject-specific covariates $\mathbf{c}_i$ and that the transformed parameters can be expressed using a linear relationship, where C_i is a matrix obtained by rearranging the covariates $\mathbf{c}_i$:

$$\phi_i = h(\psi_i) = C_i\mu + \eta_i, \tag{2}$$

so that the model parameters can be expressed as:

$$\psi_i = h^{-1}(\mu, \mathbf{c}_i, \eta_i) = h^{-1}(\phi_i). \tag{3}$$

We further assume that the vector of random effects η_i follows a multinormal distribution: $\eta \sim \mathcal{N}(0, \Omega)$. In the current version of **saemix**, h must be one of the following: the identity function (normal distribution for ψ), the logarithmic function (log-normal distribution for ψ), or the logit or probit transformations. Finally, the residual error model can be one of constant ($g = a$), proportional ($g = bf(\psi_i, x_{ij})$), combined ($g = a + bf(\psi_i, x_{ij})$), or exponential. The exponential model assumes that $y_{ij} > 0$ for all i, j and $y_{ij} = f(\psi_i, x_{ij})e^{g(\psi_i,x_{ij})\epsilon_{ij}}$, where $g = a$ (corresponding to an homoscedastic variance after log-transformation). We denote by σ the parameters of the residual error model ($\sigma = \{a, b\}$).

The unknown set of parameters in the model defined by (1) are $\theta = (\mu, \text{vec}(\Omega), \sigma)$, where $\text{vec}(\Omega)$ is the vector of the parameters of the variance-covariance matrix Ω.

2.2. Parameter estimation

In the context of maximum likelihood, we are interested in the estimates of θ which maximize the likelihood of the observations $\ell(\theta; \mathbf{y})$. Assuming the different subjects are independent, ℓ can be written as a product of the individual likelihoods, which are obtained as an integral over the distribution of the individual parameters $\mathcal{D}$:

$$\ell(\theta; \mathbf{y}) = \prod_{i=1}^{N} \ell(\theta; \mathbf{y}_i) = \prod_i p(\mathbf{y}_i|\theta) = \prod_i \int_{\mathcal{D}} p(\mathbf{y}_i|\eta_i, \theta)p(\eta_i|\theta)d\eta_i. \tag{4}$$

This likelihood has no analytical expression when f is nonlinear. To get around this issue, two main approaches can be used. The first approach involves a linearization of the model (Lindstrom and Bates 1990) or of the likelihood (Wolfinger 1993) to obtain a tractable expression, while the second approach uses numerical (Pinheiro and Bates 1995) or stochastic (Wei and Tanner 1990) approximations to compute the likelihood. Once an expression has been computed, the resulting likelihood is then maximized either through standard minimization algorithms like the quasi-Newton algorithms, or through EM algorithms (Dempster *et al.* 1977), where the unknown individual parameters are treated as missing data. In nonlinear mixed effect models, the E-step at the iteration k of the EM algorithm consists in computing the conditional expectation of the complete log-likelihood $Q_k(\theta) = \mathsf{E}(\log p(y, \psi; \theta)|y, \theta_{k-1})$ and the M-step consists in computing the value θ_k that maximizes $Q_k(\theta)$. Following Dempster *et al.* (1977) and Wu (1983), the EM sequence (θ_k) converges to a stationary point of the observed likelihood (i.e., a point where the derivative of ℓ is 0) under general regularity conditions.

In cases where the regression function f does not linearly depend on the random effects, the E-step cannot be performed in a closed-form. The stochastic approximation version of the standard EM algorithm, proposed by Delyon *et al.* (1999), consists in replacing the usual E-step of EM by a stochastic procedure. At iteration k of SAEM, the algorithm proceeds as follows:

- *Simulation-step* : draw $\psi^{(k)}$ from the conditional distribution $p(\cdot|y; \theta_k)$.
- *Stochastic approximation* : update $Q_k(\theta)$ according to

$$Q_k(\theta) = Q_{k-1}(\theta) + \gamma_k(\log p(y, \psi^{(k)}; \theta) - Q_{k-1}(\theta)). \tag{5}$$

- *Maximization-step* : update θ_k according to

$$\theta_{k+1} = \operatorname{Arg}\max_{\theta} Q_k(\theta).$$

In the simulation step, the $\psi^{(k)}$ are simulated at each iteration via an Markov chain Monte Carlo (MCMC) procedure from the current conditional distribution of the individual parameters. The same procedure can also be used after the algorithm has converged to obtain the conditional distribution of the individual parameters, from which we can compute the conditional modes, the conditional means and the conditional standard deviations. In the stochastic approximation step, the sequence of γ_k controls the convergence of the SAEM algorithm. It should be a decreasing sequence converging to 0 at a rate slower than 1 over the iteration number (Delyon *et al.* 1999). In practice, γ_k is set equal to 1 during the first K_1 iterations to let the algorithm explore the parameter space without memory and to converge quickly to a neighborhood of the maximum likelihood estimator, and the stochastic approximation is performed during the final K_2 iterations where $\gamma_k = 1/(k - K_1 + 1)$, ensuring the almost sure convergence of the estimator. Also, the step size in the first few initial iterations (6 by default) is set to 0, as we are not interested in computing Q during the run-in sequence.

The SAEM algorithm has been implemented in the MATLAB software under the name **Monolix** (Kuhn and Lavielle 2005). It has been shown theoretically to converge to a maximum of the likelihood of the observations under very general conditions (Delyon *et al.* 1999). To avoid local maxima, a simulated annealing step is implemented in a first series of iterations to ensure proper exploration of the parameter space, which confers a certain robustness with respect to the choice of initial parameter values to the SAEM algorithm. In practical applications, the SAEM algorithm implemented in **Monolix** has been shown to be effective and fast (Girard and Mentré 2005; Plan *et al.* 2012).

2.3. Estimation of the log-likelihood

The SAEM algorithm uses a stochastic approximation of the log-likelihood during the iteration phase, and thus does not compute directly the log-likelihood. The likelihood in Equation 4 can be computed at the end of the optimization process by one of several approaches.

A first possibility is to approximate (4) by the likelihood of the Gaussian model deduced from the nonlinear mixed effects model after linearization of the function f around the predictions of the individual parameters $(\psi_i, 1 \leq i \leq N)$, essentially using the same approximation proposed by Lindstrom and Bates (1990).

A second possibility is to use a Monte Carlo approach, to obtain an estimate without any model approximation. The integral $\ell(\theta; \mathbf{y})$ can be approximated via an *importance sampling* integration method (IS), using the conditional distribution $p(\psi|\mathbf{y};\theta)$ which is obtained using the empirically estimated conditional mean $\mathsf{E}(\psi_i|\mathbf{y}_i;\widehat{\theta})$ and conditional variance $\mathsf{VAR}(\psi_i|\mathbf{y}_i;\widehat{\theta})$ of ψ_i for each subject $i = 1, 2, \ldots, N$. The quality of the approximation depends on the estimates of the conditional mean and variances of the individual distributions.

A third possibility (GQ) is to use numeric integration as opposed to stochastic integration. Gauss-Hermite quadrature methods use a fixed set of K_{GQ} ordinates (called nodes) and weights $(x_k, w_k)_{k=1,\ldots,K_{GQ}}$ to approximate the likelihood function. As for importance sampling, the quality of the approximation depends on the estimates of $\mathsf{E}(\psi_i|\mathbf{y}_i;\widehat{\theta})$ and $\mathsf{VAR}(\psi_i|\mathbf{y}_i;\widehat{\theta})$.

The linearized approximation to the log-likelihood is the same approximation used in the linearization-based estimation methods, but is computed at the maximum likelihood estimates obtained by SAEM. It is fast to compute, but becomes increasingly inaccurate as the nonlinearity of the model increases. The two other methods provide respectively stochastic and numerical approximations to the true log-likelihood, which should be asymptotically equivalent when increasing the number of nodes (with Gaussian quadrature) and the number of samples (with importance sampling). In practice, both these approximations have been shown to carry a significant computational burden for large number of random effects (Pinheiro and Bates 1995). In **saemix**, we have implemented fixed Gaussian quadrature with 12 nodes by default, and the number of nodes can be set to any value between 1 and 25 through the `nnodes.gq` argument, while the number of samples for the importance sampling method may be tuned by the `nmc.is` argument. Adaptive Gaussian quadrature (AGQ) could be used to improve the approximation, as Pinheiro and Bates (1995) suggested AGQ to be more computationally efficient than IS and GQ, but is not yet implemented in **saemix**.

2.4. Estimation of standard errors

To assess the parameter estimates which have been obtained through the SAEM algorithm, we can consider the standard errors associated with the estimates. Estimates of these standard errors can be obtained as the inverse of the Fisher information matrix $I(\widehat{\theta}) = -\partial^2_\theta \log \ell(\widehat{\theta}; \mathbf{y})$ (see for instance Walter and Pronzato 2007).

As with the likelihood defined in (4), the Fisher information matrix of the nonlinear mixed effect model has no closed-form solution. Approximations to the Fisher information matrix have been proposed in the optimal design context by Mentré and others (Mentré, Mallet, and Baccar 1997; Retout, Mentré, and Bruno 2002). In **saemix** we compute the Fisher information matrix by linearization of the function f around the conditional expectation of the individual Gaussian parameters ($\mathsf{E}(\psi_i|\mathbf{y};\hat{\theta}), 1 \leq i \leq N$). The resulting model is Gaussian, and its Fisher information matrix is a block matrix (no correlations between the estimated fixed effects and the estimated variances). We compute the gradient of f numerically. Alternatively, the Louis principle could be used to compute the Fisher information matrix through a stochastic approximation that does not involve a linearization of the model (Louis 1982).

2.5. Model evaluation

Model evaluation is a vital part of model building. Detecting model deficiencies can help guide the next model to be tested. Basic model diagnostics compare model point predictions and observations, while more sophisticated diagnostics involve simulations under the model (Brendel *et al.* 2006; Comets, Brendel, and Mentré 2008, 2010). Other elements to assess model performance include predictive abilities on external datasets.

Computing predictions requires estimates of subject-specific parameters. In nonlinear mixed effect models, two sets of predictions are generally considered. Population predictions use the estimates of the population parameters and the individual design, typically dose regimen and regressors in PK/PD, while individual predictions are obtained using the individual parameters. The individual parameters (ψ_i) are obtained by estimating the individual normally distributed parameters (ϕ_i) and deriving the estimates of (ψ_i) using the transformation $\psi_i = h(\phi_i)$. These estimates are obtained through $p(\phi_i|\mathbf{y}_i;\hat{\theta})$, the conditional distribution of ϕ_i for $1 \leq i \leq N$, where $\hat{\theta}$ denotes the estimated value of θ computed with the SAEM algorithm. The MCMC procedure used in the SAEM algorithm is used to estimate these conditional distributions, and obtain for each subject i the conditional mode (or maximum a posteriori) $m(\phi_i|\mathbf{y}_i;\hat{\theta}) = \operatorname{Arg\,max}_{\phi_i} p(\phi_i|\mathbf{y}_i;\hat{\theta})$, the conditional mean $\mathsf{E}(\phi_i|\mathbf{y}_i;\hat{\theta})$, and the conditional standard deviation $sd(\phi_i|\mathbf{y}_i;\hat{\theta})$. The number of iterations of the MCMC algorithm used to estimate the conditional mean and standard deviation is adaptively chosen to maintain the sequence of empirical means and standard deviations within a ρ_{mcmc}-confidence interval during L_{mcmc} iterations.

3. Overview of the package

3.1. Package structure

The **saemix** package implements the SAEM algorithm in R using S4 object-oriented programming. The main function in the package is the `saemix()` function, which estimates the population parameters of a nonlinear mixed effect model. This function requires two mandatory arguments, the (structural and statistical) model, a '`saemixModel`' object, and the data, a '`saemixData`' object created by the `saemixData()` function. An optional list of settings can be passed on as a list. The syntax of the `saemix()` function is simply:

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

where the `control` argument is optional.

The model object is created by a call to the `saemixModel()` function, with the following main arguments:

```
saemixModel(model, psi0, description, error.model, transform.par,
  fixed.estim, covariate.model, covariance.model, omega.init, error.init,
  ...)
```

The first two arguments are mandatory, while all the others have default values.

`model`: Name of an R 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 for syntax).

`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) rows 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).

`transform.par`: The distribution for each parameter (`0` = normal, `1` = log-normal, `2` = probit, `3` = logit).

`fixed.estim`: Whether parameters should be estimated (`1`) or fixed to their initial estimate (`0`).

`covariate.model`: A matrix giving the covariate model, with `0` for parameters to be estimated and `1` for parameters set to 0. Defaults to diagonal (only variance terms are estimated, and covariances are assumed to be zero).

`covariance.model`: A square matrix of size equal to the number of parameters in the model, giving the variance-covariance matrix of the model.

`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.

`error.init`: A vector of size 2 giving the initial values of a and b in the error model.

The data object is created by a call to the `saemixData()` function:

```
saemixData(name.data, header, sep, na, name.group, name.predictors,
  name.response, name.X, name.covariates, units, ...)
```

The first argument is the only mandatory argument, and gives the name of the dataset. All others have default values, provided the columns of the dataset have names which can be identified by the program, although the user is encouraged to provide the names of the group, predictors and response columns for safety:

`name.data`: Name of the dataset (can be a character string giving the name of a file on disk or the name of a dataset in the R session).

`header`: Whether the dataset/file contains a header.

`sep`: The field separator character.

`na`: A character vector of the strings which are to be interpreted as `NA` values.

`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 modeled as a function of predictor(s) x.

`name.covariates`: Name (or number) of the column(s) containing the covariates, if present (otherwise missing).

`name.X`: (used in plots) 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.

Examples of how to use these functions are provided in Section 4.

The third argument in the call to `saemix()`, `control`, is optional; when given, it is a list containing options like the directory in which to save graphs and results, or the seed used to initialize the random number generator. The list of options which can be set is described in detail in the help files provided with the package.

The `saemix()` function returns an S4 object of class '`saemixObject`'. This object contains the following slots:

`data`: The data object.

`model`: The model object.

`results`: The results object.

`options`: A list of options containing variables controlling the algorithm.

`prefs`: A list of graphical preferences, applied to plots.

Many methods have been programmed for '`saemixObject`' objects, including `summary`, `print` and `plot` methods, as well as functions to extract likelihoods, fitted values and residuals. Other functions are specific to **saemix**, and are used to compute the likelihood (`llis.saemix`, `llgq.saemix`), estimate and sample from the conditional distribution of individual parameters (`conddist.saemix`), compute standard errors of estimation using the linearized Fisher information matrix (`fim.saemix`), and simulate observations and parameters (`simul.saemix`). All the functions are documented in the online help.

3.2. Settings

Once a fit has been performed, the resulting object contains two named lists, which contain settings for running the algorithms and graphical preferences which are used mostly for plots. A detailed description can be found in the user documentation.

Algorithmic settings

These settings are passed on to the `saemix()` function through a list. All the elements in this list are optional and will be set to default values when running the algorithm if they are not specified by the user prior to the fit as elements of the `control` list. They include: (i) settings for the SAEM algorithm (the sequence of step sizes γ, the number of burning iterations K_b); (ii) settings for the MCMC algorithm (the number of Markov chains L, the numbers m_1, m_2, m_3 and m_4 of iterations in the Hasting-Metropolis algorithm, the probability of acceptance ρ for kernel $q^{(3)}$ and $q^{(4)}$); (iii) settings for the algorithm estimating the conditional distribution of the (ϕ_i) (the width of the confidence interval ρ_{mcmc} and the number of iterations L_{mcmc}); (iv) settings for the simulated annealing algorithm (the coefficients τ_1 and τ_2 defining the decrease of the temperature and the number of iterations K_{sa}); (v) settings for the algorithms to compute the likelihood (the Monte Carlo number M for the importance sampling, number of quadrature points as well as the width of each integral for the Gaussian quadrature algorithm); (vi) which modeling steps to perform after the estimation of the population parameters (estimation of individual parameters, estimation of the standard errors, estimation of different approximations to the likelihood).

User preferences

A list of default graphical parameters is stored in the `prefs` elements of the '`saemixObject`' object returned after a fit. These parameters control colors, line and symbol types and sizes, axes labels, titles, etc. and can be used to supersede default values in the `plot` functions. Any of these settings can be changed directly in the list, and will then affect all future plots, but can also be set on the fly when calling the `plot` function, then affecting only the plot being created.

4. Examples

The objective of this section is to show how to use the **saemix** package through several examples. In the first example, we will show the main features of a typical longitudinal dataset used also in other modeling software, the orange tree dataset. This very simple example will allow us to explain the structure of the data object and to create a simple logistic model that we can fit with `saemix()`. We then show the main output from the package. In the second example, we will consider a well-known pharmacokinetic example, giving an example of a covariate model, and we will use this example to showcase the diagnostic plots provided in the package. In the third example, we will study the statistical performances of the SAEM algorithm in terms of parameter estimation through a simulation study, by applying **saemix** to an E_{max} model for dose-response data, and we show the influence of tuning the parameters in the algorithm.

In the following we will assume that the **saemix** package has been loaded:

```
R> library("saemix")
```

4.1. Orange trees

The data

The orange tree data was originally presented by Draper and Smith (1981) and was used by Pinheiro and Bates (1995) to demonstrate the **nlme** package. It is now available in the `datasets` package distributed with R, and it is also distributed with other statistical programs, most notably SAS and **WinBUGS** (Lunn, Thomas, Best, and Spiegelhalter 2000). The dataset contains measurements of the circumference at chest height of 5 orange trees, measured at seven time points during a period of 4 years. The data is available in R under the name `Orange`.

The data object is created through the function `saemixData`. We need to specify the name of the dataframe, as well as the columns containing the grouping factor (indicating the subject), the predictor(s) and the response. Here, the grouping factor is `Tree`, and the number of the tree is given in the first column, while the second contains the age of the tree (the predictor) and the third its circumference (the response).

```
R> data("Orange", package = "datasets")
R> head(Orange)
```

```
  Tree  age circumference
1    1  118            30
2    1  484            58
3    1  664            87
4    1 1004           115
5    1 1231           120
6    1 1372           142
```

```
R> orange <- saemixData(name.data = Orange, name.group = "Tree",
+    name.predictors = "age", name.response = "circumference",
+    units = list(x = "Days", y = "mm"))
```

```
Using the object called Orange in this R session as the data.
The following SaemixData object was successfully created:

Object of class SaemixData
    longitudinal data for use with the SAEM algorithm
Dataset Orange
    Structured data: circumference ~ age | Tree
    Predictor: age (Days)
```

The units element is optional, but if given, will be used to label the plots. Note that it is also possible to give the number of the columns instead of the names, as in:

```
R> orange <- saemixData(name.data = Orange, name.group = 1,
+    name.predictors = 2, name.response = 3)
```

and that there is also some automatic name recognition built into the function (see the second example in Section 4.2).

A plot of the data can be obtained by a call to the `plot` function, yielding the graph shown in Figure 1.

```
R> plot(orange)
```

Modeling

Different mathematical functions can be used to describe growth curves. Pinheiro and Bates (1995) used the following three-parameters logistic model to predict $f(x)$, the circumference of the tree at age x:

$$f(x) = \frac{\lambda_1}{1 + e^{-\dfrac{(x-\lambda_2)}{\lambda_3}}}. \tag{6}$$

They assumed an additive error model with constant variance ($\epsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$). Since the dataset contained only 5 subjects, they assumed that only the intercept λ_1 varied across trees, and they set a normal distribution for λ_1 so that $\lambda_{1i} \sim \mathcal{N}(\mu_1, \omega_1^2)$. The individual vector of parameters is then $\lambda_i = (\lambda_{1i}, \lambda_2, \lambda_3)$.

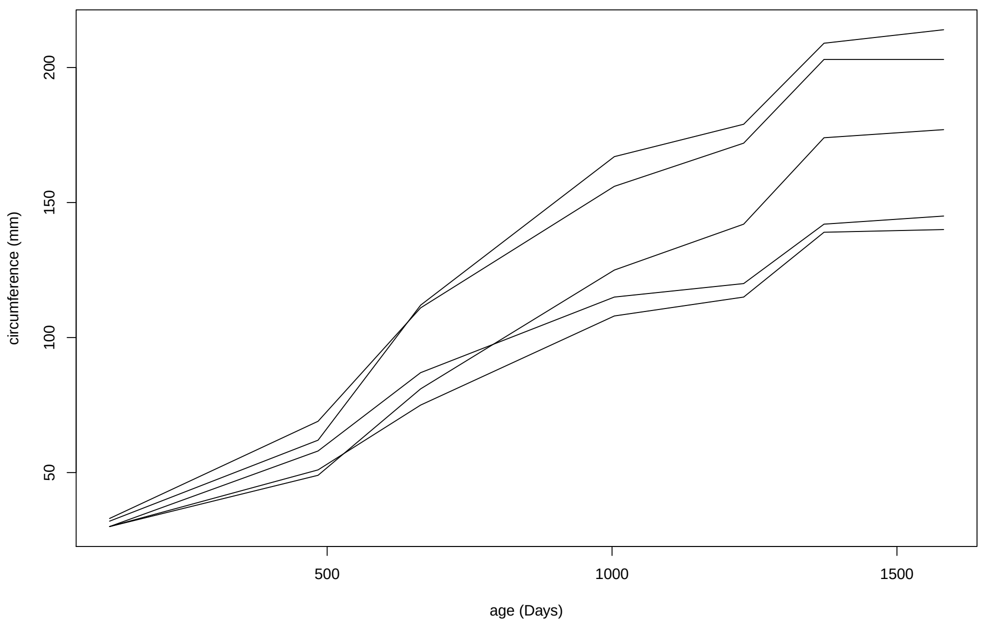


Figure 1: Orange tree data: Tree circumference (in mm) as a function of time (in days since 1968/12/31).

To set up this model in **saemix**, we first define a general model function. This function must have 3 arguments named `psi` (assumed to be a matrix with the number of columns equal to the number of parameters in the model, here 3), `id` (assumed to be a vector of indices matching observation number with subject index) and `xidep` (assumed to be a matrix with as many columns as predictors, here 1). The three arguments passed to the function will be generated automatically from the model and data object within the **saemix** code. The function must return a vector of predictions, the length of which is equal to the number of rows in the predictor `xidep`. The following code snippet shows how to define the function f for the orange tree model as a **saemix** model function:

```
R> logistic.model <- function(psi, id, xidep) {
+    age <- xidep[, 1]
+    lambda1 <- psi[id, 1]
+    lambda2 <- psi[id, 2]
+    lambda3 <- psi[id, 3]
+    resp <- lambda1 / (1 + exp( -(age - lambda2) / lambda3))
+    return(resp)
+  }
```

The user only needs to change the computation of the response, and adjust the number of predictors and parameters. More examples of model functions can be found in the documentation online, as well as in the other examples (see Sections 4.2 and 4.3).

We then build the statistical model through the function `saemixModel`:

```
R> orange.model <- saemixModel(model = logistic.model,
+    description = "Logistic growth", psi0 = matrix(c(200, 800, 400),
+    ncol = 3, byrow = TRUE, dimnames = list(NULL, c("lambda1", "lambda2",
+    "lambda3"))),  covariance.model = matrix(c(1, 0, 0, 0, 0, 0, 0, 0, 0),
+    ncol = 3, byrow = 3))

The following SaemixModel object was successfully created:

Nonlinear mixed-effects model
  Model function:  Logistic growth
function (psi, id, xidep)
{
    age <- xidep[, 1]
    lambda1 <- psi[id, 1]
    lambda2 <- psi[id, 2]
    lambda3 <- psi[id, 3]
    resp <- lambda1/(1 + exp(-(age - lambda2)/lambda3))
    return(resp)
}
  Nb of parameters: 3
      parameter names:  lambda1 lambda2 lambda3
      distribution:
     Parameter Distribution Estimated
[1,] lambda1      normal       Estimated
[2,] lambda2      normal       Estimated
[3,] lambda3      normal       Estimated
  Variance-covariance matrix:
     lambda1 lambda2 lambda3
lambda1    1    0    0
lambda2    0    0    0
lambda3    0    0    0
  Error model: constant , initial values: a = 1
    No covariate in the model.
    Initial values
             lambda1 lambda2 lambda3
Pop.CondInit  200  800  400
```

Note that in this model, we only estimate a variability for λ_1, the other parameters are the same for all subjects. Since we do not specify an initial value for the variance-covariance matrix, the default initialization is used: a diagonal matrix, the diagonal elements of which are set to 1 for parameters with a log-normal, logit or probit distribution, and to the square of the initial value (set in argument `psi0`) for parameters with a normal distribution. Since the algorithm is stochastic, non-zero values must be set even for parameters without individual variability for the algorithm to simulate samples in the run-in phase.

Parameter estimation

To run `saemix()`, we pass the model and data arguments to the function. A list of options may be specified: in the code below, we set the seed for the random number generator, and we specify that we do not want graphs and results to be saved as they would be by default.

```
R> opt <- list(seed = 94352514, save = FALSE, save.graphs = FALSE)
R> orange.fit <- saemix(orange.model, orange, opt)
```

The parameters are then estimated. The output includes a summary of the model and data, followed by the numerical results, which include parameter estimates, their standard errors and several statistical criteria. By default, the likelihood is computed both by linearization and by importance sampling, and the corresponding Akaike (AIC) and Schwarz (BIC) information criteria are computed. In this very simple model, both are practically identical. For each parameter estimated in the model, estimates of the standard error are reported, as an absolute value (SE) and relative to the estimate, as a coefficient of variation (% CV). In the present case, all the fixed parameters are well estimated, with coefficients of variation less than 10%, and the standard deviation of parameter λ_1 (measuring its variability) is estimated to be about 33, with an estimation error somewhat higher (65%) reflecting the limited information available with only 5 subjects in the dataset.

```
...
-----------------------------------
----          Results          ----
-----------------------------------
----------------------------------------------------
-----------------  Fixed effects  ------------------
----------------------------------------------------
     Parameter Estimate SE CV(%)
[1,] lambda1   196.2    16  8.2
[2,] lambda2   748.3    38  5.1
[3,] lambda3   362.7    28  7.7
[4,] a           7.9     1 12.9
----------------------------------------------------
-----------  Variance of random effects  -----------
----------------------------------------------------
        Parameter      Estimate SE  CV(%)
lambda1 omega2.lambda1 1043     673 65
----------------------------------------------------
------  Correlation matrix of random effects  ------
----------------------------------------------------
               omega2.lambda1
omega2.lambda1 1
----------------------------------------------------
---------------  Statistical criteria  -------------
----------------------------------------------------
Likelihood computed by linearisation
      -2LL =  263.4706
      AIC  =  273.4706
      BIC  =  271.5178

Likelihood computed by importance sampling
      -2LL =  263.48
```

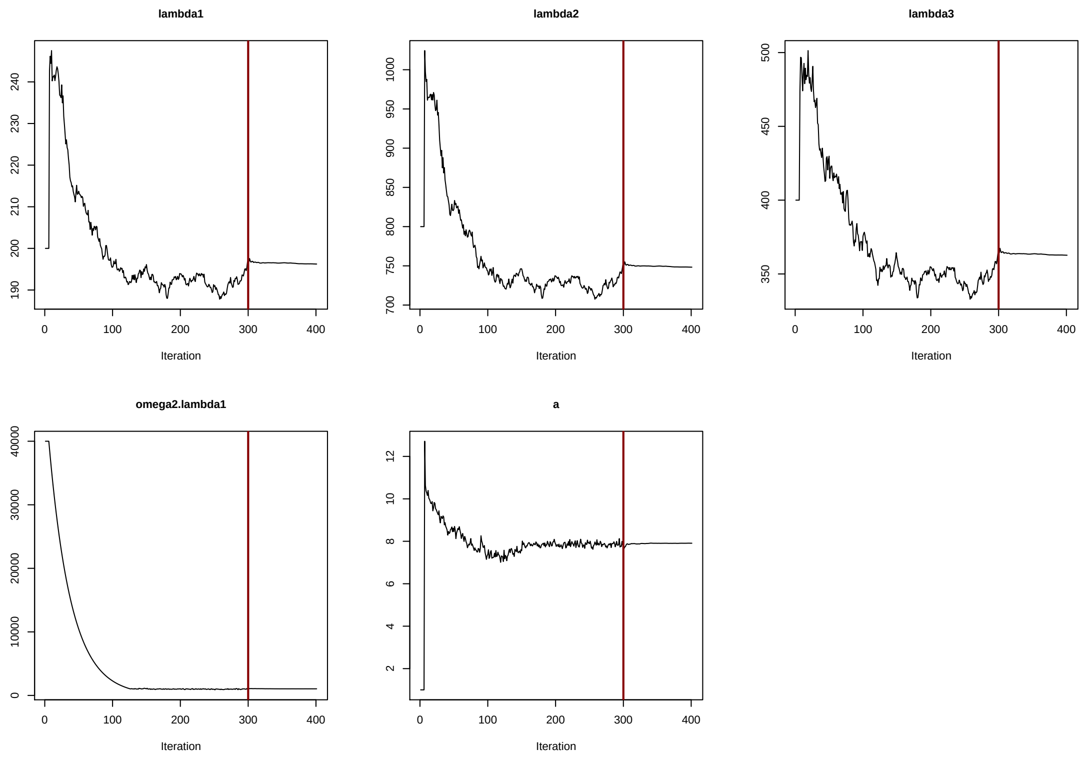


Figure 2: Convergence plots for the orange tree data.

```
AIC  =  273.48
BIC  =  271.5272
```

During the fit, the progression of convergence is assessed through plots, shown in Figure 2. The vertical line in the plots correspond to K_1 iterations, and delineates two phases in the algorithm. During the first K_1 iterations, the algorithm explores freely the parameter space, while during the second phase (K_2 iterations), the step size γ_k decreases slowly to ensure parameter convergence. The individual parameters of each subject are estimated during the second phase (K_2 iterations), and used for the diagnostics. These initial estimates can be further refined after the fit through the `conddist.saemix()` function.

Finally, we can check that the model is able to describe the observed data by producing plots of the individual fits:

```
R> plot(orange.fit, plot.type = "individual", smooth = TRUE)
```

this code adds the `smooth = TRUE` argument to use more time-points on the x-axis for the plot, and computes the prediction at each time-point using the estimated individual parameters for each tree. The result is shown in Figure 3 and illustrates a very good fit of the model for all subjects.

Corresponding code for `nlme` *and* `nlmer`

The same model can be fit to the orange data using either `nlmer` from **lme4** or `nlme` from **nlme**. The following code, adapted from the online help for `nlme`, loads the two packages and

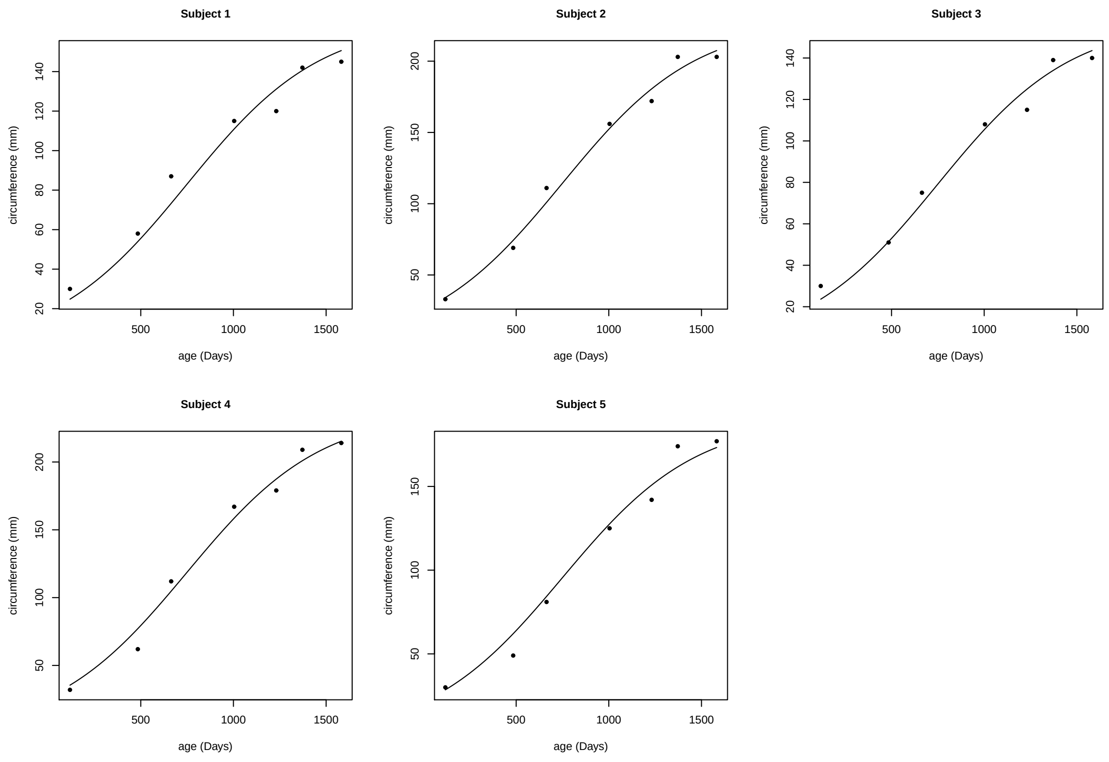


Figure 3: Plots showing the individual predictions (solid line) overlaid on the observations (black dots) for each tree in the dataset.

defines the model function as a user-defined function. It is also possible to use the built-in `SSlogis` function, but the purpose of this paragraph is to show how models in **saemix** relate to other modeling packages to facilitate switching from one to the other. The same function can be used for both `nlmer` and `nlme`, but `nlmer` needs the gradients to be defined in the function through the `deriv` function:

```
R> library("nlme")
R> library("lme4")
R> modelform <- ~ lambda1 / (1 + exp( -(age - lambda2) / lambda3))
R> nlmer.orangemod <- deriv(modelform, namevec = c("lambda1", "lambda2",
+    "lambda3"), function.arg = c("age", "lambda1", "lambda2", "lambda3"))
R> startvec <- c(lambda1 = 200, lambda2 = 800, lambda3 = 400)
```

The following code performs the fit using `nlmer`:

```
R> nlmer.orangefit <- nlmer(circumference ~ nlmer.orangemod(age, lambda1,
+    lambda2, lambda3) ~ lambda1 | Tree, Orange, start = startvec,
+    random = lambda1)
```

The following code performs the fit using `nlme`, using a slightly different syntax to specify the structure of the fixed and random effects:

```
R> nlme.orangefit <- nlme(circumference ~ nlmer.orangemod(age, lambda1,
+    lambda2, lambda3), fixed = lambda1 + lambda2 + lambda3 ~ 1,
+    random = lambda1 ~ 1, data = Orange, start = startvec)
```

The output from the `nlmer` fit is given by:

```
R> summary(nlmer.orangefit)

Nonlinear mixed model fit by maximum likelihood  ['nlmerMod']
Formula: circumference ~ nlmer.orangemod(age, lambda1, lambda2, lambda3) ~
  lambda1 | Tree
   Data: Orange

     AIC      BIC   logLik deviance df.resid
   273.1    280.9   -131.6    263.1       30

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.9170 -0.5421  0.1754  0.7116  1.6820

Random effects:
 Groups   Name    Variance Std.Dev.
 Tree     lambda1 1001.49  31.646
 Residual           61.51   7.843
Number of obs: 35, groups:  Tree, 5

Fixed effects:
        Estimate Std. Error t value
lambda1   192.05      15.58   12.32
lambda2   727.90      34.44   21.14
lambda3   348.07      26.31   13.23

Correlation of Fixed Effects:
        lambd1 lambd2
lambda2 0.384
lambda3 0.362  0.762
Warning messages:
1: In vcov.merMod(object, use.hessian = use.hessian) :
  variance-covariance matrix computed from finite-difference Hessian is
not positive definite: falling back to var-cov estimated from RX
2: In vcov.merMod(object, correlation = correlation, sigm = sig) :
  variance-covariance matrix computed from finite-difference Hessian is
not positive definite: falling back to var-cov estimated from RX
```

The output from the `nlme` fit is given by:

```
R> summary(nlme.orangefit)
```

```
Nonlinear mixed-effects model fit by maximum likelihood
  Model: circumference ~ nlmer.orangemod(age, lambda1, lambda2, lambda3)
 Data: Orange
       AIC      BIC    logLik
  273.1693 280.9461 -131.5847

Random effects:
 Formula: lambda1 ~ 1 | Tree
         lambda1 Residual
StdDev: 31.48254 7.846255

Fixed effects: lambda1 + lambda2 + lambda3 ~ 1
           Value Std.Error DF  t-value p-value
lambda1 191.0455  16.15380 28 11.82666       0
lambda2 722.5357  35.14849 28 20.55666       0
lambda3 344.1529  27.14659 28 12.67757       0
 Correlation:
        lambd1 lambd2
lambda2 0.375
lambda3 0.354  0.755

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-1.9147537 -0.5351318  0.1436489  0.7309596  1.6615020

Number of Observations: 35
Number of Groups: 5
```

In this example, the estimates of the parameters as well as the statistical criteria are very close across the three packages.

4.2. Theophylline pharmacokinetics

The data

This is again a well-known dataset, distributed in **NONMEM**, **Monolix** and R, for fitting nonlinear mixed effect models, and is a classic example of pharmacokinetic data. The data was collected in a study by Upton of the kinetics of the anti-asthmatic drug theophylline and analyzed by Boeckmann *et al.* (1994). Twelve subjects were given oral doses of the anti-asthmatic drug theophylline, then serum concentrations (in mg/L) were measured at 11 time points over the next 25 hours. In package **saemix**, we removed the data at time 0 to avoid some unexplained non-zero values in a supposedly single-dose study; in addition we transformed the doses to body doses in mg instead of doses by weight mg/kg as in the original dataset, and we added a dummy variable for gender with values 0 and 1 to illustrate fitting a categorical covariate model. The resulting dataset is available under the name `theo.saemix` in the **saemix** package.

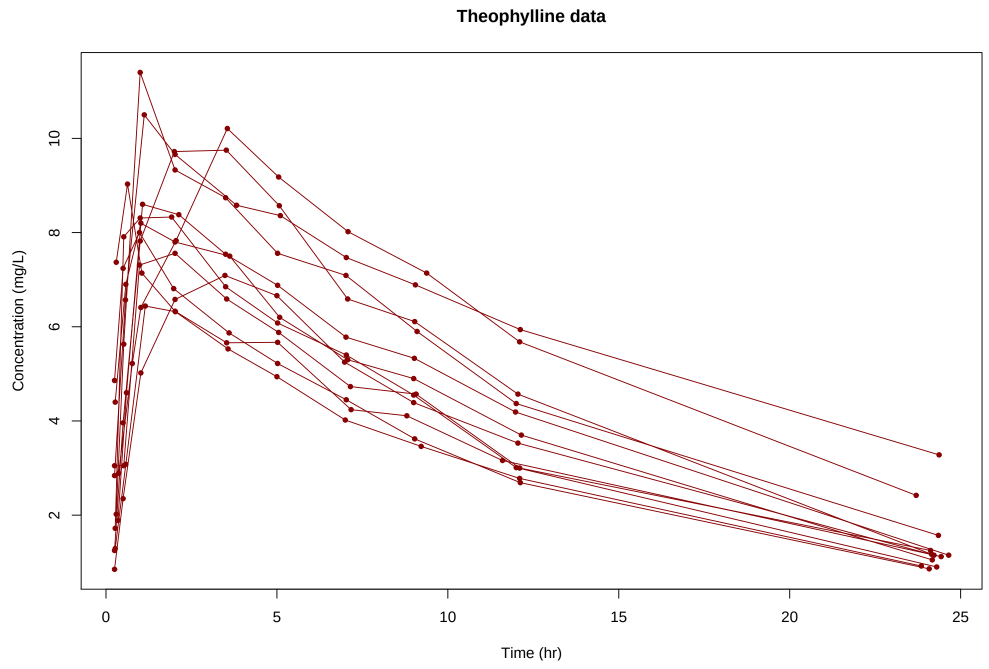


Figure 4: Concentration of theophylline versus time in 12 subjects.

The data is shown in Figure 4. In the following code, we use the dataset from the package create the ‘`saemixData`’ object:

```
R> data("theo.saemix", package = "saemix")
R> theo.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")
R> plot(theo.data, type = "b", col = "DarkRed", main = "Theophylline data")
```

The data includes 2 predictors, `Dose` and `Time`; we use the `name.X = "Time"` argument to specify which of the predictor will be used as the independent variable in the plots. We read the two covariates in the data object, and we specify units for the independent and dependent variables, as well as for the covariates. The last line in this code is used to plot the data, changing a few options such as plot type, color and title by passing these arguments to the `plot` function.

Modeling

These data were analyzed in Davidian and Giltinan (1995) and Pinheiro and Bates (1995) using a two-compartment open pharmacokinetic model. Subject i receives an initial dose D_i at time 0. Serum concentrations y_{ij} measured at time t_{ij} are modeled by a first-order one compartment model, according to the following equation:

$$y_{ij} = \frac{D_i k_{\mathrm{ai}} k_{\mathrm{ei}}}{CL_i(k_{\mathrm{ai}} - k_{\mathrm{ei}})} \left(e^{-k_{\mathrm{ai}} t_{ij}} - e^{-k_{\mathrm{ei}} t_{ij}} \right) + \epsilon_{ij}, \tag{7}$$

where CL_i is the clearance of subject i, k_{ai} is the absorption rate constant, k_{ei} is the elimination rate constant and is expressed as a function of CL_i and the volume of distribution V_i as $k_{\text{ei}} = \frac{CL_i}{V_i}$. For subject i, the vector of regression (or design) variables is $x_{ij} = (D_i, t_{ij})$, and the vector of individual parameters is $\theta_i = (k_{\text{ai}}, CL_i, V_i)$. k_{ai}, CL_i and V_i are assumed to be independent log-normal random variables (corresponding to $h(x) = \ln(x)$), and we assume a relationship between the clearance and the subject's body weight BW_i:

$$\begin{aligned} \ln(k_{\text{ai}}) &= \mu_1 + \eta_1, \\ \ln(V_i) &= \mu_2 + \eta_2, \\ \ln(CL_i) &= \mu_3 + \beta_{WT,CL}\,\text{BW}_i + \eta_3. \end{aligned} \tag{8}$$

We used a simple homoscedastic error model where $\mathsf{VAR}(\epsilon_{ij}) = a^2$.

The following code is used to write a model function, and to create a '`saemixModel`' object:

```
R> model1cpt <- function(psi, id, xidep) {
+    dose <- xidep[, 1]
+    tim <- xidep[, 2]
+    ka <- psi[id, 1]
+    V <- psi[id, 2]
+    CL <- psi[id, 3]
+    k <- CL / V
+    ypred <- dose * ka / (V * (ka - k)) * (exp(-k * tim) - exp(-ka * tim))
+    return(ypred)
+ }
R> theo.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))
```

The covariate model is specified as a matrix, and is reported in the model object as:

```
  Covariate model:
     ka V CL
[1,]  0 0  1
[2,]  0 0  0
```

This indicates that the first covariate in the dataset (`Weight` in our example) will be assumed to have a relationship with parameter CL. In the code, `psi0` now includes a second line, which is used to specify initial values for the parameter-covariate relationships. The same initial values would be used for all covariates, but we could specify different starting values for each parameter-covariate relationship by adding more lines to the matrix `psi0`. Alternatively, we can also let the program use default values of 0 by letting `psi0` be a one-line matrix as in the orange tree example.

Parameter estimation

As previously, we fit this model using the `saemix()` function:

```
R> opt <- list(save = FALSE, save.graphs = FALSE)
R> theo.fit <- saemix(theo.model, theo.data, opt)
```

```
...
-----------------------------------
----          Results          ----
-----------------------------------
----------------------------------------------------
-----------------  Fixed effects  ------------------
----------------------------------------------------
     Parameter       Estimate SE     CV(%) p-value
[1,] ka               1.5786  0.2947  18.7 -
[2,] V               31.6605  1.4322   4.5 -
[3,] CL               1.5521  0.9683  62.4 -
[4,] beta_Weight(CL)  0.0082  0.0089 108.3 0.18
[5,] a                0.7429  0.0569   7.7 -
----------------------------------------------------
-----------  Variance of random effects  -----------
----------------------------------------------------
   Parameter Estimate SE     CV(%)
ka omega2.ka 0.368    0.1668 45
V  omega2.V  0.017    0.0096 57
CL omega2.CL 0.065    0.0324 50
----------------------------------------------------
------  Correlation matrix of random effects  ------
----------------------------------------------------
          omega2.ka omega2.V omega2.CL
omega2.ka 1         0        0
omega2.V  0         1        0
omega2.CL 0         0        1
----------------------------------------------------
---------------  Statistical criteria  -------------
----------------------------------------------------
Likelihood computed by linearisation
      -2LL =  343.427
      AIC  =  359.427
      BIC  =  363.3063

Likelihood computed by importance sampling
      -2LL =  344.8205
      AIC  =  360.8205
      BIC  =  364.6997
----------------------------------------------------
```

Again fixed effects are well estimated, except for the fixed effect of weight on clearance, which is estimated with a variability of 108%.

Covariate model

We can test whether the covariate effect is significant. In nonlinear mixed effect models, standard tests are the Wald test and the log-likelihood ratio test (LRT).

The Wald test compares the value of a parameter estimate $\hat{\Psi}^{(k)}$ to the standard error of estimation $\widehat{SE}(\hat{\Psi}^{(k)})$ through a χ^2 statistic with one degree of freedom:

$$W = \left(\frac{\hat{\Psi}^{(k)}}{\widehat{SE}(\hat{\Psi}^{(k)})} \right)^2 \sim \chi^2_1. \tag{9}$$

This is the test performed in the output above for the fixed effects involved in a covariate model, and here we see that in this example, the fixed effect representing the influence of weight on *CL* is not significant (p = 0.18, NS according to the Wald test), in keeping with the large estimated *SE*.

The LRT compares two nested models through the difference in twice the log-likelihood (LL). Assuming the more parsimonious model M_1 has p parameters and the larger model M_2 $p+q$ parameters, asymptotically the difference in $-2LL$ follows a χ^2 statistic with q degree of freedom:

$$Z = -\left(2LL_1 - 2LL_2\right)^2 \sim \chi^2_q. \tag{10}$$

In **saemix**, the likelihood can be computed by three different approximations. Two of these are computed by default with the standard settings of the algorithm. To compute also the likelihood by Gaussian quadrature, we use the `llgq.saemix()` function:

```
R> theo.fit <- llgq.saemix(theo.fit)
R> theo.fit
```

This adds the log-likelihood computed by Gaussian quadrature to the object:

```
...
Likelihood computed by Gaussian quadrature
      -2LL =  344.7868
       AIC =  360.7868
       BIC =  364.666
```

To test the covariate model using the LRT, we first need to fit a model without the covariate, then we compute the LRT statistic. Here we use the log-likelihoods computed by linearisation, but in this example we get similar p values with all approximations.

```
R> theo.model.base <- 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))
R> opt <- list(save = FALSE, save.graphs = FALSE)
R> theo.base <- saemix(theo.model.base, theo.data, opt)
R> ll1 <- theo.base["results"]["ll.lin"] * (-2)
R> ll2 <- theo.fit["results"]["ll.lin"] * (-2)
R> 1 - pchisq(ll1 - ll2, 1)
```

```
[1] 0.4365312
```

Model selection

Statistical tools for model selection include information criteria and hypothesis tests such as the Wald test and likelihood ratio test (LRT) for nested models. Non-nested models can be compared through the Akaike criterion (AIC) and Schwarz's information criterion, also called Bayes information criterion or BIC. These two statistics are reported in the output for each likelihood computed. Their value is given by:

$$\begin{aligned} AIC &= -2LL + 2p, \\ BIC &= -2LL + p\ \ln(n_{\mathrm{tot}}), \end{aligned} \tag{11}$$

where p is the number of parameters in the model and n_{tot} the total number of observations.

AIC and BIC are justified based on asymptotic criteria (Burnham and Anderson 2002), that is, when the sample size increases and the model dimension stays fixed. In the context of mixed effect models, the effective sample size is not clearly defined and the formula for calculating the BIC differs from software to software. The penalty using n_{tot} is implemented in the R package **nlme** and SPSS procedure `MIXED`, while the number of subjects, N, is used in **Monolix** and SAS `PROC NLMIXED`. An optimal BIC penalty based on two terms proportional to $\log N$ and $\log n_{\mathrm{tot}}$ and that is consistent with the random effect structure in a mixed effect model was recently proposed by Delattre, Poursat, and Lavielle (2014).

Other criteria have been proposed and studied for nonlinear mixed effect models (see the review in Bertrand, Comets, Chenel, and Mentré 2012); they can be computed from the estimate of the log-likelihood, which can be accessed through elements of the results object. For instance, the log-likelihood computed by importance sampling can be obtained as:

```
R> logLik(theo.fit)

LL by is "-172.41 (df=8)"
```

We should also note that a conditional AIC (cAIC) has been proposed by Vaida and Blanchard (2005) for linear mixed effect models. Its theoretical properties have been investigated by Greven and Kneib (2010) and implemented for **lme4** in an R package. In nonlinear mixed effect models, small sample corrections have been proposed for the Wald test by Bertrand *et al.* (2012), who also suggest using permutation tests to control type I error inflation in model selection with the LRT and the Wald test.

Model diagnostics

Diagnostics are destined to evaluate model properties and help guide model building. A number of diagnostic graphs are produced automatically when running `saemix()`, and are output by a call to the `plot()` function applied to the object resulting from the fit. Specific plots can also be created using the `plot.type` argument to the function. When called individually, each graph can be tailored to user preferences through options as in the previous example (Figure 4). The next code snippet shows how to obtain specific graphs, given in Figures 5 to 8. The first graph, in Figure 5, displays the individual fits for the first 4 subjects in the theophylline example, including a smoothed prediction line, and changing the color of the line and the plotting symbol. A logarithmic scale is used for the y-axis. Figure 6 plots the observations versus the predictions, adding the unity line to assess deviations. Figure 7 assesses

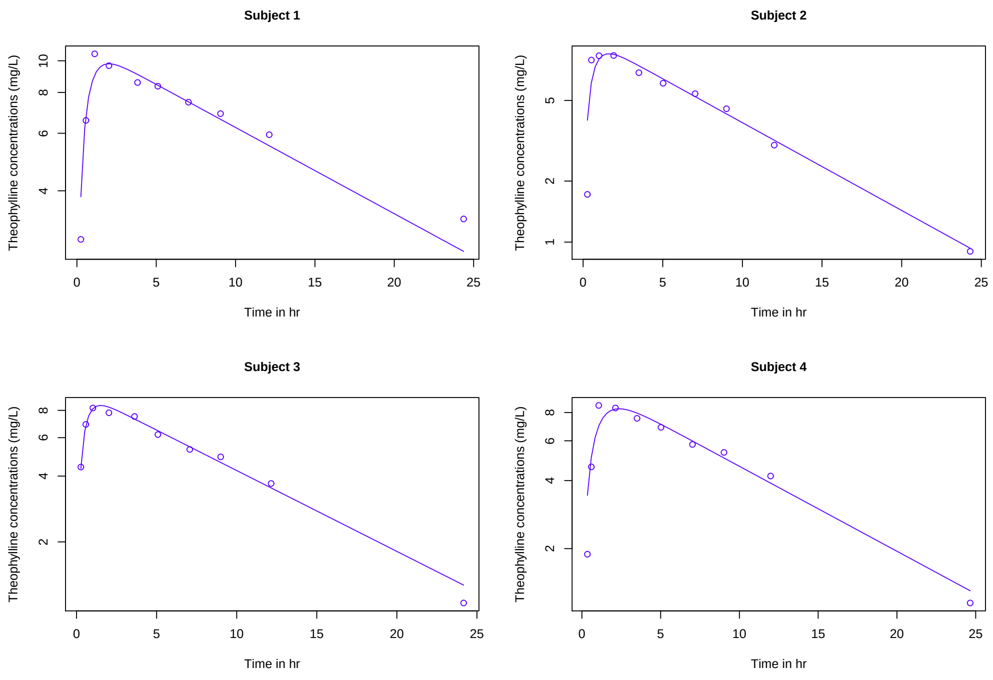


Figure 5: Individual plots for the first 4 subjects in the study, with different options.

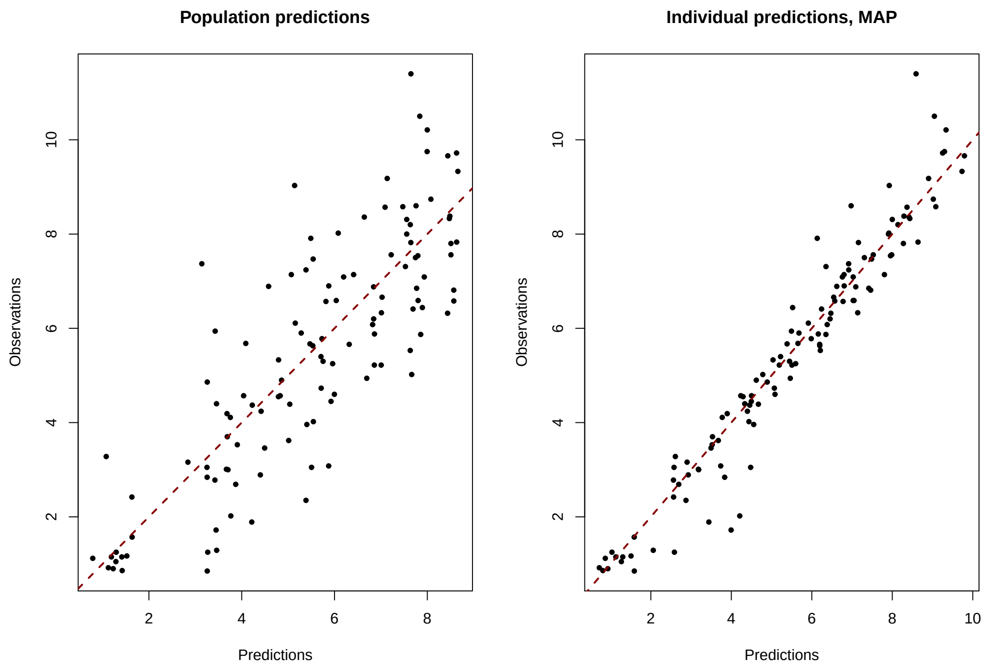


Figure 6: Observations versus predictions for the theophylline data, with population predictions on the left and individual predictions on the right.

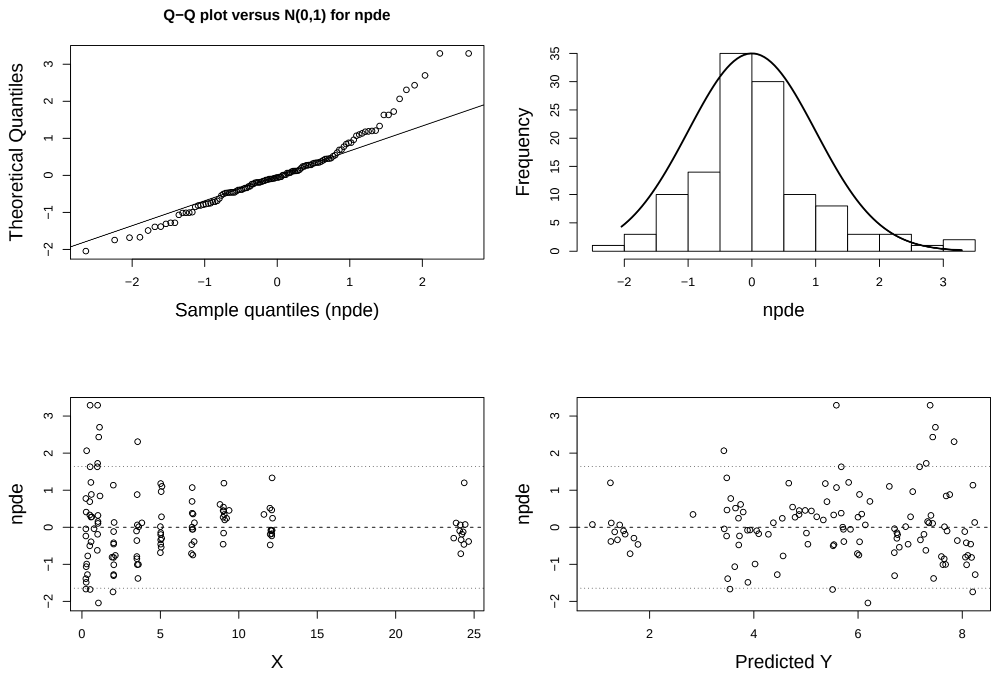


Figure 7: Residual diagnostics based on npde for the theophylline data.

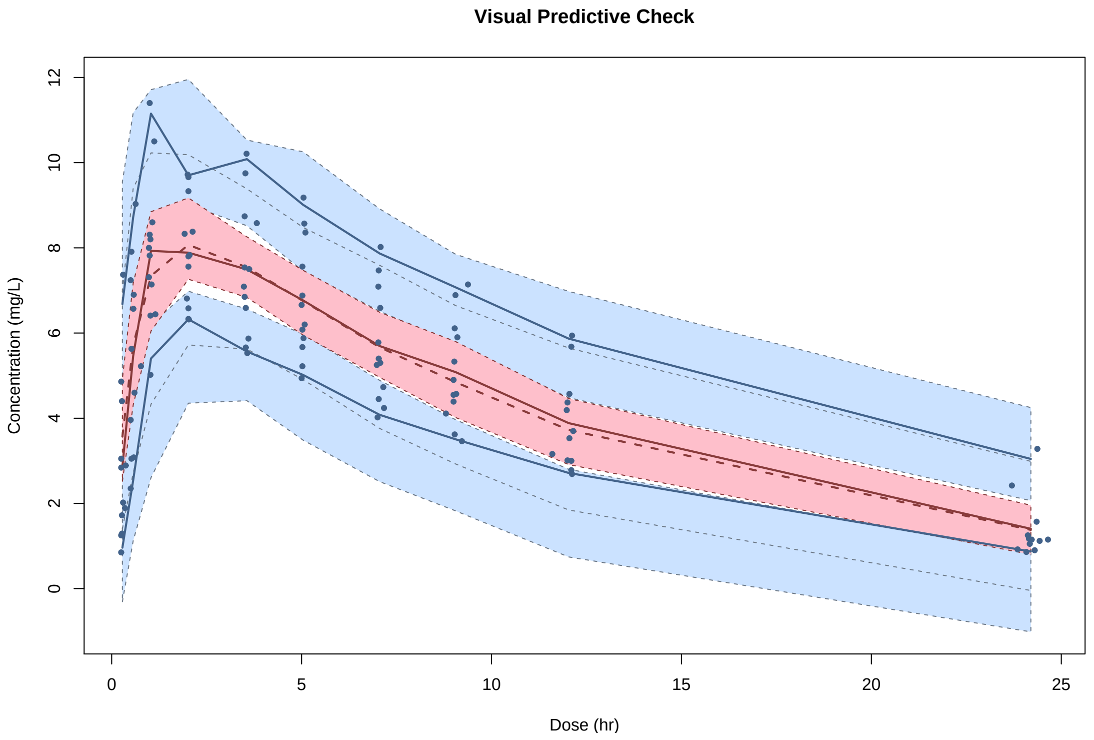


Figure 8: VPC for the theophylline data.

the distribution of npde, a residual adapted to nonlinear mixed effect models (Brendel *et al.* 2006; Comets *et al.* 2010), while Figure 8 gives the visual predictive check (VPC).

```
R> par(mfrow = c(2, 2))
R> plot(theo.fit, plot.type = "individual.fit", new = FALSE, ilist = 1:4,
+    smooth = TRUE, ylog = TRUE, pch = 1, col = "Blue", xlab = "Time in hr",
+    ylab = "Theophylline concentrations (mg/L)")
R> plot(saemix.fit, plot.type = "observations.vs.predictions")
R> plot(saemix.fit, plot.type = "npde")
R> plot(saemix.fit, plot.type = "vpc")
```

Overall, these plots show good model adequacy, and individual profiles indicate that the model is able to reproduce observed data reasonably well. Some points to note may be the departure of the npde from the theoretical normal distribution seen in Figure 7, as well as a slight tendency in the plots of npde versus time. The VPC plot in Figure 8 shows that the model predicts the median evolution very well; the variability may be slightly overestimated but with only 12 subjects in the dataset this may be expected.

Corresponding code for `nlme` *and* `nlmer`

Again, we can cast the same model for use in `nlme` and `nlmer`. For `nlme`, the relationship between parameters and covariates is set in the `fixed` argument within the function call:

```
R> startvec.ke <- c(lKe = -2.99, lKa = 0.5, lCl = 0.75, lClSex = 0)
R> nform <- ~Dose * exp(lKe + lKa - lCl) * (exp(-exp(lKe) * Time) -
+    exp(-exp(lKa) * Time)) / (exp(lKa) - exp(lKe))
R> nlme.theomod.ke <- deriv(nform, namevec = c("lKe", "lKa", "lCl"),
+    function.arg = c("Dose", "Time", "lKe", "lKa", "lCl"))
R> nlme.fit.sex <- nlme(Concentration ~ nlme.theomod.ke(Dose, Time, lKe,
+    lKa, lCl), data = groupedData(Concentration ~ Time | Id,
+    data = theo.saemix), fixed = list(lKe ~ 1, lKa ~ 1, lCl ~ Sex),
+    random = pdDiag(lKe + lKa + lCl ~ 1), start = startvec.ke)
```

For `nlmer` on the other hand, the model function needs to be rewritten to explicitly include the covariate effects, as follows:

```
R> nform.sex <- ~ Dose * exp(lKe + lKa - lCl - lClSex * Sex) *
+    (exp(-exp(lKe) * Time) - exp(-exp(lKa) * Time)) / (exp(lKa) - exp(lKe))
R> nlmer.theomod.ke.sex <- deriv(nform.sex,
+    namevec = c("lKe", "lKa", "lCl", "lClSex"),
+    function.arg = c("Dose", "Time", "lKe", "lKa", "lCl", "lClSex", "Sex"))
R> nlmer.fit.sex <- nlmer(Concentration ~ nlmer.theomod.ke.sex(Dose, Time,
+    lKe, lKa, lCl, lClSex, Sex) ~ (lKe | Id) + (lKa | Id) + (lCl | Id),
+    data = theo.saemix, start = list(nlpars = startvec.ke))
```

In both cases, the model had to be reparameterized in terms of log-parameters (*lKe*, *lKa* and *lCl*) as both packages only handle normal distributions for the random effects. In addition, both `nlme` and `nlmer` fail to converge when the model is reparameterized with the volume of distribution (V) instead of the elimination rate constant Ke, which is equal to CL/V. Although we can reparameterize the model as we did here, it is worth noting that this changes the mixed effect model, because it modifies the structure of the interindividual variability. Another point to note is that for `nlmer`, a new function needs to be created for each parameter-covariate relationship that the user wishes to test.

4.3. Evaluating the performance of saemix

Data and model

In this section, we use simulated data to evaluate the performance of **saemix**. The simulated datasets were generated by Plan *et al.* (2012) in a paper comparing different software packages to estimate parameters in nonlinear mixed effect models, and were obtained directly from the authors of that paper and are included in the supplementary material.

The model used for this simulation is a sigmoid $\mathrm{E}_{\max}$ model, a standard model in dose-response studies where the effect of a drug in response to a dose d, $\mathrm{E}(d)$, is a sigmoid function corresponding to the following equation:

$$\mathrm{E}(d) = \mathrm{E}_0 + \mathrm{E}_{\max} \frac{d_i^\gamma}{d_i^\gamma + \mathrm{ED}_{50}^\gamma}. \tag{12}$$

This model involves 4 parameters, the initial effect E_0, the maximum effect $\mathrm{E}_{\max}$, the concentration at which half the maximum effect is achieved ED_{50} and the sigmoidicity factor γ which controls the nonlinearity of the model through the curvature. Interindividual variability was modeled through a log-normal distribution for all parameters, except for γ which was assumed to be the same for all subjects (no IIV). A correlation was simulated between $\mathrm{E}_{\max}$ and ED_{50}. In Plan *et al.* (2012), 3 values of γ were tested, 1, corresponding to the $\mathrm{E}_{\max}$ model, and 2 and 3, involving increasing amounts of nonlinearity, along with two residual error models, additive or proportional error. In the present work, we focus on the scenarios with $\gamma = 3$ with constant variance σ^2 (scenarios R3A for the rich and S3A for the sparse design), but full results on the 8 scenarios simulated in Plan *et al.* (2012) are available on request. The parameters used in the simulation are given in Table 1.

Simulation study

The design of the simulation study mimicked that of a clinical trial including 100 individuals and investigating four dose levels (0, 100, 300, and 1000 mg). Two sampling designs were

Parameter	Value	Parameter	Value
E_0 (–)	5	$\omega^2_{\mathrm{E}_0}$	0.09
$\mathrm{E}_{\max}$ (–)	30	$\omega^2_{\mathrm{E}_{\max}}$	0.49
ED_{50} (mg)	500	$\omega^2_{\mathrm{ED}_{50}}$	0.49
γ (–)	3	$\mathsf{COV}(\mathrm{E}_0, \mathrm{E}_{\max})$	0.245
σ (–)	2		

Table 1: Parameters used in the simulation for the $\mathrm{E}_{\max}$ model (see Plan *et al.* 2012).

.

Settings	Default		Tuned	
Initial parameters	`TRUE`	`FALSE`	`TRUE`	`FALSE`
E_0	0.43 (4.80)	−0.04 (4.74)	−0.14 (4.37)	−0.23 (4.28)
E_{max}	1.34 (17.52)	4.54 (17.56)	1.47 (13.68)	2.95 (13.54)
ED_{50}	−0.56 (13.03)	1.70 (13.07)	−0.24 (10.53)	0.88 (10.84)
γ	3.35 (10.04)	0.97 (9.32)	1.79 (6.94)	0.54 (6.46)
σ	0.59 (7.96)	0.40 (7.26)	−0.48 (6.40)	−0.32 (6.37)
$\omega^2_{E_0}$	−5.15 (37.39)	−3.74 (35.40)	0.90 (28.38)	0.32 (27.93)
$\omega^2_{E_{max}}$	−0.53 (21.45)	3.30 (27.86)	−1.54 (20.88)	0.23 (21.14)
$\omega^2_{ED_{50}}$	−2.98 (36.71)	3.58 (37.47)	−1.86 (31.20)	1.27 (30.08)
COV(E_{max}, ED_{50})	−1.47 (55.93)	10.69 (65.08)	−0.88 (48.28)	4.85 (48.72)
ρ	−4.90 (32.64)	0.71 (33.45)	−2.66 (27.60)	0.69 (26.85)

Table 2: Relative bias (RRMSE, %) in the scenario with rich sampling.

Settings	Default		Tuned	
Initial parameters	`TRUE`	`FALSE`	`TRUE`	`FALSE`
E_0	2.50 (7.97)	0.96 (8.31)	0.31 (6.26)	−2.03 (6.75)
E_{max}	−4.28 (22.69)	38.18 (68.24)	−1.20 (21.75)	33.50 (46.78)
ED_{50}	−2.72 (19.96)	34.88 (59.25)	−1.86 (18.72)	29.52 (43.07)
γ	12.17 (30.54)	−12.18 (23.26)	6.58 (19.89)	−16.93 (21.18)
σ	3.80 (16.90)	6.05 (19.58)	−0.99 (11.32)	−0.16 (11.30)
$\omega^2_{E_0}$	−17.50 (79.45)	−19.60 (86.07)	3.27 (54.95)	5.58 (60.81)
$\omega^2_{E_{max}}$	−5.78 (37.85)	21.06 (67.68)	−0.24 (33.97)	29.38 (62.72)
$\omega^2_{ED_{50}}$	−18.51 (47.84)	29.30 (72.66)	−6.36 (42.43)	43.91 (71.99)
COV(E_{max}, ED_{50})	−31.22 (82.42)	51.60 (139.77)	−5.84 (71.81)	80.19 (131.59)
ρ	−35.20 (67.43)	2.91 (59.09)	−11.90 (50.82)	25.35 (44.63)

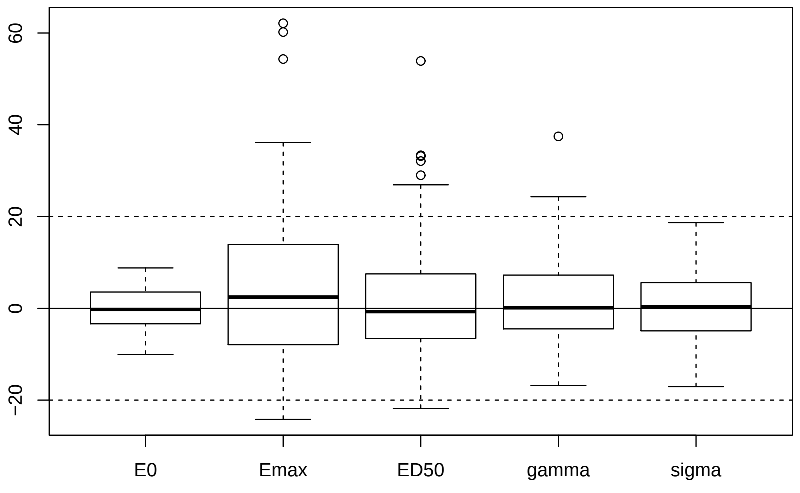
Table 3: Relative bias (RRMSE, %) in the scenario with sparse sampling.

evaluated, a rich design (R) in which all individuals were sampled at the four dose levels, and a sparse design (S) where each individual was randomly allocated to only two of the four dose levels.

We evaluated the influence of the starting values and of the tuning parameters on the performance of **saemix**. Initial parameter estimates were either set to the values used in the simulation (`TRUE`) or to different values (`FALSE`), where the fixed parameters were multiplied by 2 while the variability parameters were set to 0.1. We also used either the default settings of the algorithm, or tuned the settings by increasing the number of chains to 5 and increasing the number of iterations to 300 and 150 respectively in the burn-in and convergence phases of the algorithm. Combining starting values and tuning parameters, we therefore performed 4 successive parameter estimations for each dataset. In all settings, we used the same random seed for all runs.

Evaluation

The performance of both algorithms was evaluated by computing the relative bias and root mean square error (RMSE) over the 100 simulations, using the simulation parameters as true



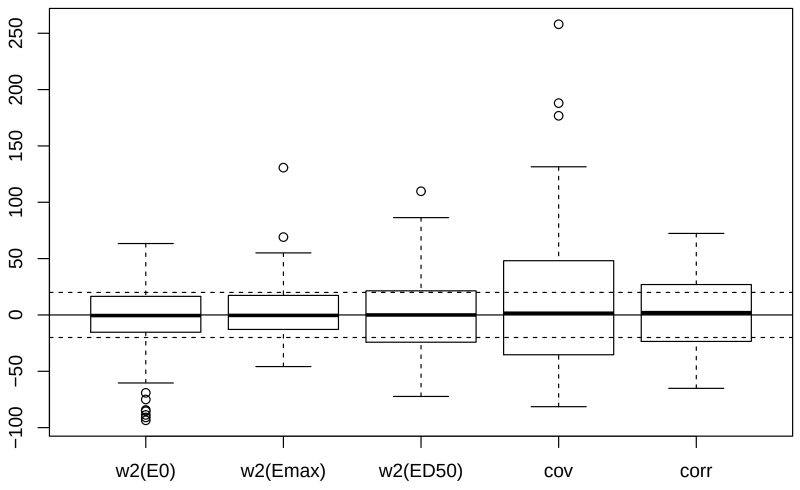


Figure 9: Bias over 100 simulations for the parameters estimated in the rich design. The solid line signals an absence of bias ($y = 0$) and the two dotted lines are plotted at ± 20% bias.

values. Denoting Ψ_0^k the true value of parameter Ψ^k, and $\hat{\Psi}_l^k$ the value estimated in the lth simulated dataset, the relative bias for this parameter was given by

$$RB(\Psi^k) = \sum_{l=0}^{100} \frac{\hat{\Psi}_l^k - \Psi_0^k}{\Psi_0^k} \tag{13}$$

and the relative RMSE by

$$RRMSE(\Psi^k) = \sum_{l=0}^{100} \frac{(\hat{\Psi}_l^k - \Psi_0^k)^2}{{\Psi_0^k}^2}. \tag{14}$$

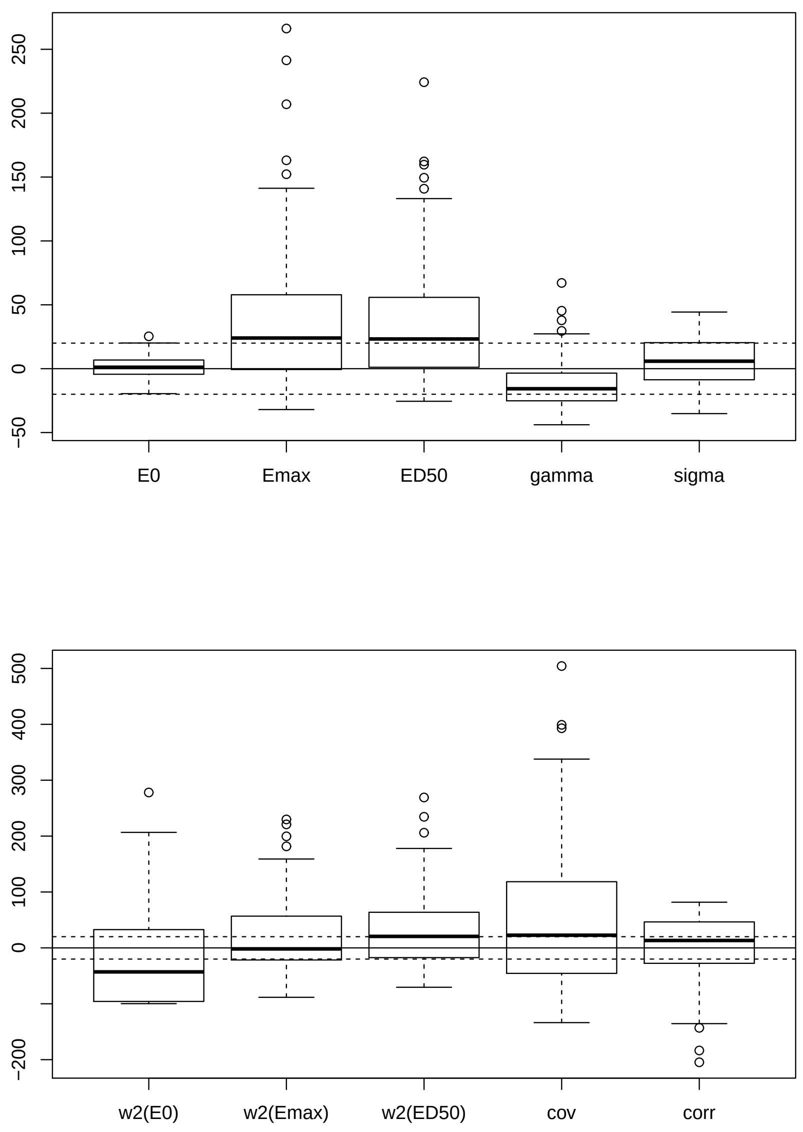


Figure 10: Bias over 100 simulations for the parameters estimated in the sparse design. The solid line signals an absence of bias ($y = 0$) and the two dotted lines are plotted at $\pm$ 20% bias.

Results

Tables 2 and 3 show the relative bias and root mean square error obtained for both methods in the two scenarios, in the four settings.

In the rich sampling scenario, all parameters are estimated without bias (less than 5% relative bias) irrespective of whether initial parameters were set to the true value or not, or whether the algorithm was tuned. The relative root mean square errors are less than 20% for the fixed effects, and between 20 and 50% for the variability parameters. Figure 9 shows a boxplot of the biases observed in the 100 datasets, when running the algorithm with the default settings and false starting values: For most of the datasets and all parameters except the covariance the bias is less than 20% in the majority of the datasets.

In the sparse sampling scenario (Table 3) on the other hand, some bias is apparent for several parameters: The fixed parameters ED_{50} and E_{max} are poorly estimated when the starting parameters are moved away from the true values used in the simulation, even when the number of chains and iterations is increased. Their variabilities are also poorly estimated, with a large RMSE. This is consistent with the results found in Plan *et al.* (2012). In that paper, most algorithms had difficulties estimating ED_{50} and E_{max} with the sparse design, which was quite challenging as it combined limited individual information and high model nonlinearity, and all methods showed biases over 20 to 30% on several parameters, worsening when perturbed initial values were used.

This is illustrated in Figure 10 with the same settings as in Figure 9. This time most of the datasets show bias for E_{max} and ED_{50}, and the variability between the datasets in bias is also much larger, especially for variabilities.

5. Operational characteristics

saemix is one of several packages in R that can be used to fit nonlinear mixed effect models. In this section, we first provide a comparison of **saemix** to **nlme** and **lme4** in terms of bias and precision of parameter estimates, as well as convergence. In a second part, we investigated run times when model complexity and dataset size increase to evaluate the scalability of the approach.

5.1. Comparing `saemix` to `nlme`/`nlmer`

Simulation study

The datasets from Section 4.3 have been fit using **nlme** by Plan *et al.* (2012), who report convergence for only 5% (rich design) and 16% (sparse design) of the datasets. We had the same issue when trying to fit the same data (results not shown). We attempted to use `nlmer` from **lme4** and had the same convergence problems; with `nlmer` we also had a technical issue, in that the derivative with respect to γ is not defined for a dose of 0, which required some tinkering with the data to bypass. Because we could fit so few datasets, we could not compare meaningfully the performance of **saemix** to those of the other two modelling packages. In Plan *et al.* (2012), the rate of convergence decreased as γ increased, suggesting it was related to model nonlinearity. In this section, we therefore decided to use the datasets simulated with $\gamma = 1$ and we fit them using an E_{max} model corresponding to Equation 12 with $\gamma = 1$ (fixed, not estimated). The datasets for the sparse design were derived from the datasets with a rich design (scenario R1A) by using for each dataset the same sampling times as in scenario S3A analysed in the previous section.

Initial conditions for `nlmer` were set to the true values used in the simulation, given in Table 1, as for `saemix`, except for $\mathsf{COV}(E_0, E_{max})$ and σ which could not be initialised in the other packages. As we encountered convergence issues again, we changed the initial parameter estimates within 20% of the original values; this was repeated a maximum of 10 times per dataset.

	`saemix`	`nlme`	`nlmer`
E_0	−2.51 (5.84)	0.29 (4.69)	2.12 (5.46)
E_{max}	1.09 (9.43)	−1.92 (7.98)	−2.56 (8.57)
ED_{50}	−2.54 (14.41)	−8.05 (13.48)	−6.34 (12.36)
σ	1.12 (6.86)	1.94 (6.26)	−0.95 (6.11)
$\omega^2_{E_0}$	1.09 (42.38)	7.73 (31.12)	0.76 (32.00)
$\omega^2_{E_{max}}$	−1.84 (19.21)	−8.50 (18.22)	−1.66 (18.32)
$\omega^2_{ED_{50}}$	−2.13 (37.46)	−27.60 (34.83)	−16.60 (30.95)
Failed runs	0	0	56

Table 4: Relative bias and RRMSE for the three packages in the scenario with rich sampling.

	`saemix`	`nlme`	`nlmer`
E_0	−1.33 (9.06)	−4.38 (9.90)	0.80 (7.56)
E_{max}	1.07 (16.56)	−0.24 (24.68)	−0.90 (7.88)
ED_{50}	0.42 (30.30)	−10.04 (47.48)	−4.83 (18.50)
σ	6.71 (18.71)	5.83 (17.35)	−8.42 (22.48)
$\omega^2_{E_0}$	−16.90 (80.13)	12.70 (69.36)	−6.43 (61.69)
$\omega^2_{E_{max}}$	−3.05 (42.46)	−17.68 (31.96)	−12.18 (32.46)
$\omega^2_{ED_{50}}$	−17.34 (107.92)	−37.20 (105.09)	−37.10 (58.41)
Failed runs	0	14	83

Table 5: Relative bias and RRMSE for the three packages in the scenario with sparse sampling.

Results

Tables 4 and 5 show the bias and RMSE for the parameters estimated by each of three packages for the rich and sparse scenarios respectively.

A first result concerns convergence rates: While `saemix` provided estimates for all datasets, we could not obtain parameter estimates for more than half of the datasets with `nlmer` despite retrying a number of times. With `nlme`, all datasets could be fit in the rich scenario but 14% failed with a sparse design. In terms of estimation errors, `saemix` provided unbiased estimates in the rich design, while both `nlme` and `nlmer` had difficulties to estimate ED_{50} and its variability. In the sparse scenario, both `saemix` and `nlme` tended to slightly overestimate σ and underestimate variabilities. `saemix` tended to underestimate the variability on E_0 and ED_{50}, while `nlme` strongly underestimated the variabilities on E_{max} and ED_{50}. The results of `nlmer` were not meaningful given that estimates could be obtained in only 17 datasets out of 100, but they also indicated similar issues.

These results show convergence issues with `nlmer` and to a lesser extent with `nlme`, in keeping with the findings reported in Plan *et al.* (2012), which had been obtained with a more nonlinear model. In our simplified settings, on a very simple E_{max} model, `nlme` converged more often than with high sigmoidicity, but showed more estimation bias than `saemix`. These results were obtained using **nlme** version 3-1.128 and **lme4** version 1.1-12.

5.2. Computation times

We performed two series of simulations to evaluate the scalability of the algorithm. In the

E_{max}	N subjects	n samples	Run time, 100 simulations (s)
	30	3	340
	30	6	419
	30	12	582
	100	3	654
	100	6	823
	100	12	1191
	200	3	1198
	200	6	1580
	200	12	2289
	500	3	2878
	500	6	3718
	500	12	5325
Exponential	Number of random effects		Run time, 100 simulations (s)
		2	348
		4	662
		6	1157

Table 6: Run times in seconds for different models, exploring the effect of increasing the number of samples and the number of subjects in the E_{max} model (30 to 500 subjects, 3 to 12 samples), and of increasing the number of parameters in the model (exponential models with 2, 4 and 6 parameters).

first, we investigated the effect of the size of the datasets by simulating datasets using the E_{max} model from Section 4.3, and varying the number of samples and the number of subjects. In the second, we assessed the effect of increasing model complexity on the run times. For this second simulation, we simulated three models with one, two and three exponential terms (respectively 2, 4 and 6 parameters). Each dataset was simulated assuming 100 subjects and 6 sample times, which had been optimised by the **PFIM** software (Retout, Comets, Samson, and Mentré 2007) to ensure reasonable identifiability of the most complex model. The model with two exponentials is similar to the E_{max} model in terms of model complexity. In each simulation, 100 datasets were generated, and the run times for the fit of the 100 datasets were collected through the `Rprof()` function in R, and summarised. Table 6 shows the total run times in seconds reported by `Rprof()`; the same information is displayed in Figure 11, with the graph on the left showing the evolution of run times with the number of subjects, stratified by the number of samples, and the graph on the right showing the influence of model complexity.

The values in Table 6 were obtained on a personal computer and are purely indicative, as they depend on CPU, memory and concurrent tasks. The relative values between the different settings however show clearly how the algorithm scales with increasing model complexity (roughly linearly with respect to the number of parameters to estimate), and with the size of the dataset. In Figure 11, there is a slight bend in the curve between a number of subjects equal to 30 and a number of subjects higher than 100. This is because when there are less than 50 subjects in the datasets, more chains are generated by the algorithm in order to improve model stability, increasing the number of 'individuals' considered in the algorithm. Other than that, run times are seen to increase linearly with the number of subjects and more

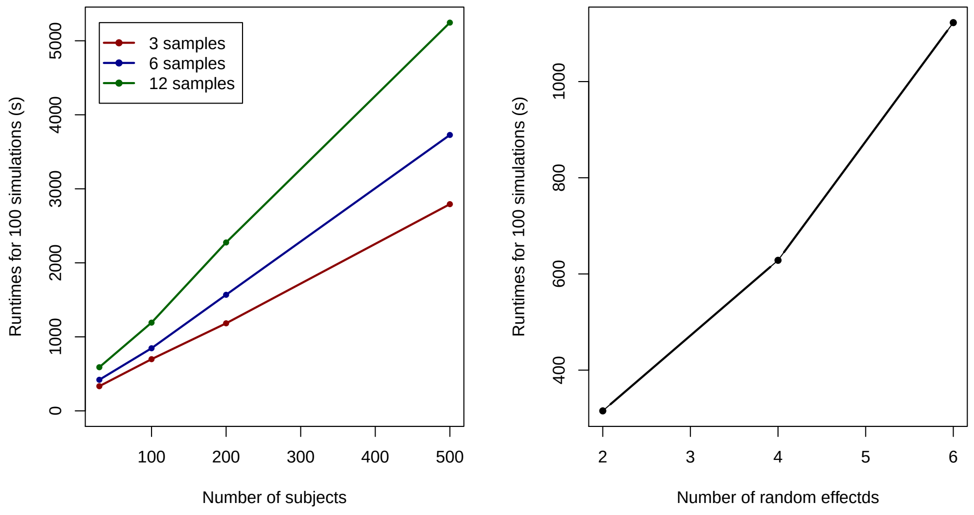


Figure 11: Evolution of run times with the number of subjects, the number of samples, and model complexity.

slowly with the number of samples. Run times also increase approximately linearly with model complexity. These results show the algorithm behaves as expected, as the number of random effects to be generated at each iteration of the algorithm depends on both model complexity and number of subjects, while the number of samples influences the time spent to compute model predictions and many intermediate computations. These profiling results will also be used to identify computational bottlenecks in further developments of **saemix**.

6. Discussion

The use of modelling and simulation in clinical drug development is now well established. Regardless of whether a single outcome is considered at the end of the study, clinical trials routinely collect longitudinal data, with each subject providing several measurements throughout the study. Longitudinal data is a staple in particular of pharmacokinetic (PK) and pharmacodynamic (PD) studies, which are a required part of a new drug application file (Lee *et al.* 2011). Nonlinear mixed effect models can help to characterise and to understand many complex nonlinear biological processes, such as biomarkers or surrogate endpoints, and are crucial in describing and quantifying the mechanisms of drug action and the different sources of variation, e.g., the interindividual variability (Ette and Williams 2007). As a result, these models have been extensively studied over the past two decades, with many developments in methods for parameter estimation, experimental design, or model evaluation. Parameter estimation can be performed through maximum likelihood (ML), restricted maximum likelihood (REML), or Bayesian approaches.

The **saemix** package provides an implementation of one of these recent estimation algorithms, SAEM, in R, to obtain maximum likelihood estimates. It offers an interesting alternative to linearisation-based estimation methods as implemented for instance in **nlme** and **lme4**. The different packages all have their own specificities and limitations. The oldest, **nlme**, is included in the base packages distributed with R, and as **lme4**, can handle multiple levels of variability. They both implement maximum and restricted maximum likelihood estimation. **lme4** also offers an approach based on adaptive-Gaussian quadrature but only when the model is linear. A limitation of `nlmer` is that there is currently no way to fit heteroscedastic error models, which are frequently used in pharmacokinetics. In practice however, `nlme` and `nlmer` often run into convergence issues, especially as model complexity increases. Both `nlme` and `nlmer` require the model to be specified in closed form; an extension **nlmeODE** has been developed to handle structural models defined by a differential equation system (Tornøe, Agersø, Jonsson, Madsen, and Nielsen 2004), but it also suffers from convergence issues. **saemix** also has some limitations in the models that can be set up. Covariates must enter the parameter model linearly (after one of the available transformations), and there is no automated treatment of categorical covariates, although this can be circumvented by users creating sets of binary covariates. In addition, for the moment **saemix** can only handle one response. Finally, the current version of the package is better suited to models with closed-form solutions, although there is no limitations from an algorithmic point of view on the models that can be fit.

Using a system of differential equations (ODE) can be implemented quite easily within R using **deSolve** (Soetaert, Petzoldt, and Setzer 2010) for instance (see Supplementary code file `fitODE_saemix.R` for an example). This approach could be used to implement a model with Michaelis-Menten elimination, for which no closed form solution of the model exists. We however recommend against integrating ODE models with `ode()`. Indeed, run times are exceedingly long even with the theophylline example, which is a very simple first-order elimination model with only 12 subjects. These run times would in addition increase linearly with the number of subjects, quickly becoming intractable. It is also possible with **deSolve** to implement the structural model in Fortran to improve the computational time but this is beyond the scope of this paper.

The SAEM algorithm has been shown to have good statistical properties in various theoretical (Kuhn and Lavielle 2005) and practical applications (Girard and Mentré 2005; Panhard and Samson 2009; Comets, Lavenu, and Lavielle 2011). We show here that the R implementation **saemix** performs well on reference datasets used to compare different software packages by Plan *et al.* (2012). Most of the scenarios run with **nlme** showed various degrees of convergence issues in that paper, whether started from the true or from altered parameters. Estimates were obtained for only 5% of the datasets simulated under a rich design, and 16% of those simulated under a sparse design in Plan *et al.* (2012). Convergence issues were related to the nonlinearity between dose and response, with a γ of 3. Of note, FOCE in **NONMEM** or SAS did not show the same convergence problems, suggesting the R implementation of the FOCE algorithm could probably be improved (Plan *et al.* 2012). We attempted here to fit the same datasets with `nlmer` (data not shown), but encountered even more convergence problems. Note also we could not fit the original dataset directly with `nlmer`, as the gradient of the model is undefined for a dose of 0. We had to change the dataset by assuming a dose of 0.01 instead of 0 to avoid this numerical problem. **nlme** for the same datasets also showed major convergence issues. In the present paper, we therefore compared **saemix** to **nlme** and `nlmer` on a less challenging setting using an $\mathrm{E}_{\max}$ model instead of the Hill model. However, `nlmer` still failed in over half of the datasets we attempted to fit, despite trying different initial parameter estimates. **nlme** fared better in terms of convergence, but still exhibited more bias than **saemix**. Both packages were run with the default settings, which might explain in part the poor results we found here.

The SAEM algorithm involves repeated evaluations of the model function, and run times increase with the number of subjects in the dataset. One possibility to improve the speed of

saemix will be to implement a stopping criterion to move into the second phase of the algorithm when the estimates of the parameters stabilise, as has been done in **Monolix** (Lavielle 2014). A sizeable portion of the time spent in the main algorithm is devoted to evaluations of the model predictions. Model predictions could be obtained from more efficient codes in C. Other computationally intensive functions include the computation of the log-likelihood by importance sampling, which again requires repeated model evaluations. These sections could again be externalised to C in future versions of **saemix**.

saemix has a number of settings that the user can change. Most of the time these settings can be left at the default values, with the exception of the following. With small sample sizes, the number of chains (argument `nb.chains`) can be increased as the results are then averaged over the different chains and yield more stable estimates from run to run. The number of iterations (argument `nbiter.saemix`) in both burn-in and convergence phases of the algorithm can also improve convergence, as shown in Section 4.3. In practice, the convergence plots can be used to assess the stability of the parameter estimates at the end of the burn-in phase to detect whether to increase the number of iterations. The number of samples used to compute the log-likelihood by importance sampling (argument `nmc.is`) can be increased to compute a more precise estimate of the log-likelihood if the estimate does not appear stable yet in the convergence plots. Obviously, increasing the values of these settings will have an impact on run times. Finally, the robustness of the estimated parameters can be assessed by changing the random seed (argument `seed`) and initial estimates of the parameters around the final estimates. Practical illustrations are proposed in Lavielle (2014).

To use **saemix**, the user needs to define two specific objects, one containing the hierarchical data structure and the other containing the elements of the mixed effect models. The structural model is defined through a R function that the user must create, which is one of the difference with **nlme** or **lme4** which both use R formulas. Although it requires a bit more work from the user to create this function, it is clear from the examples in Section 4.2 and 4.3 that this it not very difficult. Specifying the interindividual variability in **saemix** is very straightforward: The random effects are sampled from normal distributions and the user only needs to specify the transformation for each random effect, as well as the joint covariance structure. This allows great flexibility, compared to, e.g., **nlme** where changing from a normal to a log-normal distribution requires to reparameterise the model and its derivatives in terms of log-parameters. The other advantage is a clearer separation between regressors and covariates, which are defined in the data object, and parameters, which belong to the model. This is a much more flexible structure to include for instance complex dose regimens or other inputs to the system. The structure of the '`saemixModel`' object is also more easy to interface with other outputs (C programs, specialised packages) which may not comply with R standards. The future of **saemix** will indeed be tied in with larger developments that are ongoing in the pharmacometrics community. There is currently a European consortium called DDMoRe working on developing a general language for pharmacometrics, encompassing structural models from pharmacokinetics to system biology and statistical models ranging from frequentist to Bayesian approaches (Harnisch, Matthews, Chard, and Karlsson 2013). Two authors of this paper (Comets and Lavielle) are part of this endeavour and **saemix** will be extended to support this language.

In conclusion, we have developed the R package **saemix** to estimate the parameters of nonlinear mixed effect models. We do not intend to suggest that it is better than the other modelling packages available in R, merely that it can be used to complement the existing array of estimation methods. It is very easy to use, as we show in the current paper how to cast models in **nlme** and `nlmer` in terms of **saemix**, so that it can be used to compare the results for different estimation methods and evaluate the robustness of the analysis. However, the underlying algorithm, SAEM, has been shown to have superior statistical properties and good operational performances when compared to gradient-based algorithms. We hope to extend **saemix** to more complicated models by using the MLX-TRAN language and solver, which are being externalised to C/C++ so that the models can be called and model predictions obtained from different software packages. Other extensions will include more complex data, involving simultaneous modelling of several responses, left-censored data, as well as allow the user to model the interindividual variability through other parameter distributions and write his or her own residual error model. These more advanced features are already available in the **Monolix** software, which is implemented in MATLAB and has many specific features adapted to PK/PD applications as well as an extensive library of PK/PD models and a modelling language where ODE models can be easily coded. However, the objective in developing **saemix** was not to provide a complete rewrite of **Monolix**, but to offer the SAEM estimation algorithm to the R community, to be used in conjunction with or as an alternative to other packages for nonlinear mixed effect models. As such, it has many potential applications for the analysis of longitudinal data, which are encountered for instance in agronomy, chemistry, medical trials and of course pharmacokinetics and pharmacodynamics.

Acknowledgments

We thank Elodie Plan and her co-authors Alan Maloney, France Mentré, Mats Karlsson, Julie Bertrand, for kindly providing us the datasets used in Plan *et al.* (2012) to evaluate **saemix** in Section 4.3.

References

Bates D, Mächler M, Bolker B, Walker S (2015). "Fitting Linear Mixed-Effects Models Using **lme4**." *Journal of Statistical Software*, **67**(1), 1–48. `doi:10.18637/jss.v067.i01`.

Bertrand J, Comets E, Chenel M, Mentré F (2012). "Some Alternatives to Asymptotic Tests for the Analysis of Pharmacogenetic Data Using Nonlinear Mixed Effects Models." *Biometrics*, **68**(1), 146–155. `doi:10.1111/j.1541-0420.2011.01665.x`.

Bertrand J, Comets E, Mentré F (2008). "Comparison of Model-Based Tests and Selection Strategies to Detect Genetic Polymorphisms Influencing Pharmacokinetic Parameters." *Journal of Biopharmaceutical Statistics*, **18**(6), 1084–1102. `doi:10.1080/10543400802369012`.

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

Brendel K, Comets E, Laffont C, Laveille C, Mentré F (2006). "Metrics for External Model Evaluation with an Application to the Population Pharmacokinetics of Gliclazide." *Pharmaceutical Research*, **23**(9), 2036–2049. `doi:10.1007/s11095-006-9067-5`.

Burnham KP, Anderson DR (2002). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. 2nd edition. Springer-Verlag, New York.

Comets E, Brendel K, Mentré F (2008). "Computing Normalised Prediction Distribution Errors to Evaluate Nonlinear Mixed-Effect Models: The **npde** Add-On Package for R." *Computer Methods and Programs in Biomedicine*, **90**(2), 154–166. doi:10.1016/j.cmpb.2007.12.002.

Comets E, Brendel K, Mentré F (2010). "Model Evaluation in Nonlinear Mixed Effect Models, with Applications to Pharmacokinetics." *Journal de la Société Française de Statistique*, **151**(1), 106–128.

Comets E, Lavenu A, Lavielle M (2011). "**saemix**, an R Version of the SAEM Algorithm." In *20th Meeting of the Population Approach Group in Europe, Athens, Greece*. Abstract 2173.

Comets E, Lavenu A, Lavielle M (2017). ***saemix**: Stochastic Approximation Expectation Maximization (SAEM) Algorithm*. R package version 2.1, URL https://CRAN.R-project.org/package=saemix.

Comets E, Mentré F (2001). "Evaluation of Tests Based on Individual versus Population Modelling to Compare Dissolution Curves." *Journal of Biopharmaceutical Statistics*, **11**(3), 107–123.

Davidian M, Giltinan D (1995). *Nonlinear Models for Repeated Measurement Data*. Chapman & Hall, London.

Delattre M, Poursat MA, Lavielle M (2014). "A Note on BIC in Mixed-Effects Models." *Electronic Journal of Statistics*, **8**(1), 456–475. doi:10.1214/14-ejs890.

Delyon B, Lavielle M, Moulines E (1999). "Convergence of a Stochastic Approximation Version of the EM Algorithm." *The Annals of Statistics*, **27**(1), 94–128.

Dempster AP, Laird NM, Rubin DB (1977). "Maximum Likelihood From Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society B*, **39**(1), 1–38.

Draper NR, Smith H (1981). *Applied Regression Analysis*. 2nd edition. John Wiley & Sons, New York.

Ette E, Williams P (2007). *Pharmacometrics: The Science of Quantitative Pharmacology*. John Wiley & Sons, Hoboken.

Girard P, Mentré F (2005). "A Comparison of Estimation Methods in Nonlinear Mixed Effects Models Using a Blind Analysis." In *14th Meeting of the Population Approach Group in Europe, Pamplona, Spain*.

Greven S, Kneib T (2010). "On the Behaviour of Marginal and Conditional AIC in Linear Mixed Models." *Biometrika*, **97**(4), 773–789.

Harnisch L, Matthews I, Chard J, Karlsson MO (2013). "Drug and Disease Model Resources: A Consortium to Create Standards and Tools to Enhance Model-Based Drug Development." *CPT: Pharmacometrics & Systems Pharmacology*, **2**(3), 1–3. doi:10.1038/psp.2013.10. 34.

Kuhn E, Lavielle M (2005). "Maximum Likelihood Estimation in Nonlinear Mixed Effects Models." *Computational Statistics & Data Analysis*, **49**(4), 1020–1038. doi:10.1016/j.csda.2004.07.002.

Lavielle M (2014). *Mixed Effects Models for the Population Approach – Models, Tasks, Methods and Tools*. Chapman & Hall/CRC, Boca Raton.

Lee JY, Garnett CE, Gobburu JVS, Bhattaram VA, Brar S, Earp JC, Jadhav PR, Krudys K, Lesko LJ, Li F, Liu J, Madabushi R, Marathe A, Mehrotra N, Tornøe C, Wang Y, Zhu H (2011). "Impact of Pharmacometric Analyses on New Drug Approval And Labelling Decisions: A Review of 198 Submissions Between 2000 And 2008." *Clinical Pharmacokinetics*, **50**(10), 627–635. doi:10.2165/11593210-000000000-00000.

Lindstrom MJ, Bates DM (1990). "Nonlinear Mixed Effects Models for Repeated Measures Data." *Biometrics*, **46**(3), 673–87. doi:10.2307/2532087.

Louis TA (1982). "Finding the Observed Information Matrix When Using the EM Algorithm." *Journal of the Royal Statistical Society B*, **44**(2), 226–233.

Lunn D, Thomas A, Best NG, Spiegelhalter DJ (2000). "**WinBUGS** – A Bayesian Modelling Framework: Concepts, Structure, and Extensibility." *Statistics and Computing*, **10**(4), 325–337.

Mentré F, Mallet A, Baccar D (1997). "Optimal Design in Random-Effects Regression Models." *Biometrika*, **84**(2), 429–442.

Molenberghs G, Verbeke G (2005). *Models for Discrete Longitudinal Data*. Springer-Verlag, Berlin.

Panhard X, Samson A (2009). "Extension of the SAEM Algorithm for Nonlinear Mixed Models with 2 Levels of Random Effects." *Biostatistics*, **10**(1), 121–135.

Pinheiro JC, Bates DM (1995). "Approximations to the Log-Likelihood Function in the Non-Linear Mixed-Effect Models." *Journal of Computational and Graphical Statistics*, **4**(1), 12–35. doi:10.2307/1390625.

Plan E, Maloney A, Mentré F, Karlsson MO, Bertrand J (2012). "Performance Comparison of Various Maximum Likelihood Nonlinear Mixed-Effects Estimation Methods for Dose-Response Models." *The AAPS Journal*, **14**(3), 420–432. doi:10.1208/s12248-012-9349-2.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Retout S, Comets E, Samson A, Mentré F (2007). "Design in Nonlinear Mixed Effects Models: Optimization Using the Fedorov-Wynn Algorithm and Power of the Wald Test for Binary Covariates." *Statistics in Medicine*, **26**(28), 5162–5179. doi:10.1002/sim.2910.

Retout S, Mentré F, Bruno R (2002). "Fisher Information Matrix for Non-Linear Mixed-Effects Models: Evaluation and Application for Optimal Design of Enoxaparin Population Pharmacokinetics." *Statistics in Medicine*, **21**(18), 2623–2639. doi:10.1002/sim.1041.

SAS Institute Inc (2013). *The SAS System, Version 9.4*. SAS Institute Inc., Cary. URL http://www.sas.com/.

Savic R, Mentré F, Lavielle M (2011). "Implementation and Evaluation of the SAEM Algorithm for Longitudinal Ordered Categorical Data with an Illustration in Pharmacokinetics-Pharmacodynamics." *The AAPS Journal*, **13**(1), 44–53. doi:10.1208/s12248-010-9238-5.

Soetaert K, Petzoldt T, Setzer RW (2010). "Solving Differential Equations in R: Package **deSolve**." *Journal of Statistical Software*, **33**(9), 1–25. doi:10.18637/jss.v033.i09.

The MathWorks Inc (2014). *MATLAB – The Language of Technical Computing, Version R2014b*. Natick. URL http://www.mathworks.com/products/matlab/.

Tornøe CW, Agersø H, Jonsson EN, Madsen H, Nielsen HA (2004). "Non-Linear Mixed-Effects Pharmacokinetic/Pharmacodynamic Modelling in NLME Using Differential Equations." *Computer Methods and Programs in Biomedicine*, **76**(1), 31–40. doi:10.1016/j.cmpb.2004.01.001.

Vaida F, Blanchard S (2005). "Conditional Akaike Information for Mixed-Effects Models." *Biometrika*, **92**(2), 351–370.

Walter E, Pronzato L (2007). *Identification of Parametric Models from Experimental Data*. Springer-Verlag, New York.

Wei G, Tanner M (1990). "Calculating the Content and Boundary of the Highest Posterior Density Region via Data Augmentation." *Biometrika*, **77**(3), 649–652. doi:10.2307/2337005.

Wolfinger R (1993). "Laplace's Approximation for Nonlinear Mixed Models." *Biometrika*, **80**(4), 791–795. doi:10.2307/2336870.

Wu CFJ (1983). "On the Convergence Properties of the EM Algorithm." *The Annals of Statistics*, **11**(1), 95–103.

Affiliation:

Emmanuelle Comets
CIC 1414, Laboratoire de pharmacologie expérimentale et clinique
Faculté de Médecine

2 av. Pr Léon Bernard
35 000 Rennes, France
Telephone : +33/0/2-23-23-47-65
Fax : +33/2-23-23-46-05
E-mail: emmanuelle.comets@inserm.fr
URL: http://www.saemix.biostat.fr/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
August 2017, Volume 80, Issue 3 *Submitted:* 2014-03-07
doi:10.18637/jss.v080.i03 *Accepted:* 2016-07-28