SAEMIX

Version 1.2

February 2014

Emmanuelle Comets [1,2], *Audrey Lavenu* [2], *Marc Lavielle* [3]

[1] INSERM, IAME UMR 1137, Paris, France; Université Paris Diderot, Paris, France

[2] University Rennes-I, Rennes, France; INSERM CIC 1414, Rennes, France

[3] INRIA, Saclay, France

Email: emmanuelle.comets@inserm.fr

Contents

Chapter 1

Introduction

1.1 The objectives

The objectives of SAEMIX are to perform:

1. parameter estimation for nonlinear mixed effects models
   - computing the maximum likelihood estimator of the population parameters, without any approximation of the model (linearization, quadrature approximation, ...), using the Stochastic Approximation Expectation Maximization (SAEM) algorithm,
   - computing standard errors for the maximum likelihood estimator
   - computing the conditional modes, the conditional means and the conditional standard deviations of the individual parameters, using the Hastings-Metropolis algorithm
2. goodness of fit plots
3. model selection
   - comparing several models using some information criteria (AIC, BIC)
   - testing hypotheses using the Likelihood Ratio Test
   - testing parameters using the Wald Test

The R package SAEMIX is an implementation of the Stochastic Approximation Expectation Maximization (SAEM) algorithm in R [24], developed by Kühn and Lavielle, and implemented in the MONOLIX software available in Matlab and as a standalone software for Windows and Linux [14].

The current version of the R version of SAEMIX handles only analytical functions. The following features have not yet been implemented in the R package SAEMIX, but are available in the MONOLIX software:

- categorical covariates with more than 2 categories
- models defined with differential equations
- multi-response model
- left censored data
- interoccasion variability
- prior distribution for the random effects
- complex variables, including discrete data or repeated time to events
- hidden Markov models
- mixture models
- autocorrelation of the residuals

Theoretical analysis of the algorithms used in this software can be found in [6, 8, 12, 13]. Several application of SAEM in agronomy [19], animal breeding [11] and PKPD analysis [3, 17, 27, 29, 1] have been published by several members of the Monolix group. Several applications to PKPD analysis were also proposed during the last PAGE (Population Approach Group in Europe) meetings ([20, 16, 15, 26, 28, 30] as well as a comparison of estimation algorithms [10], (http://www.page-meeting.org).

The present document describes the nonlinear mixed effects models (section 1) and the algorithms used in this package (section 2). The final section shows some examples made available in the library.

1.2 The nonlinear mixed effects model

Detailed and complete presentations of the nonlinear mixed effects model can be found in [4, 5, 23]. See also the many references therein.

We consider the following general nonlinear mixed effects model for continuous outputs:

$$y_{ij} = f(x_{ij}, \psi_i) + g(x_{ij}, \psi_i, \xi)\varepsilon_{ij} \quad , \ 1 \leq i \leq N \quad , \quad 1 \leq j \leq n_i \tag{1.1}$$

Here,

- $y_{ij} \in \mathbb{R}$ is the jth observation of subject i,
- N is the number of subjects,
- n_i is the number of observations of subject i,
- the regression variables, or design variables, (x_{ij}) are assumed to be known, $x_{ij} \in \mathbb{R}^{n_x}$,
- for subject i, the vector $\psi_i = (\psi_{i,\ell}\,;\, 1 \leq \ell \leq n_\psi) \in \mathbb{R}^{n_\psi}$ is a vector of n_ψ individual parameters:

$$\psi_i = H(\mu, c_i, \eta_i) \tag{1.2}$$

  where

  - $c_i = (c_{im}\,;\, 1 \leq m \leq M)$ is a known vector of M covariates,
  - μ is an unknown vector of fixed effects of size n_μ,
  - η_i is an unknown vector of normally distributed random effects of size n_η:

$$\eta_i \sim_{i.i.d.} \mathcal{N}(0, \Omega)$$

- the residual errors (ε_{ij}) are random variables with mean zero and variance 1,
- the residual error model is defined by the function g and some parameters ξ.

Here, the parameters of the model are $\theta = (\mu, \Omega, \xi)$. We will denote $\ell(y;\theta)$ the likelihood of the observations $y = (y_{ij}\,;\, 1 \leq i \leq n\ , 1 \leq j \leq n_i)$ and $p(y, \psi; \theta)$ the likelihood of the complete data $(y, \psi) = (y_{ij}, \psi_i\,;\, 1 \leq i \leq n\ , 1 \leq j \leq n_i)$. Thus,

$$\ell(y;\theta) = \int p(y, \psi; \theta)\ d\psi.$$

1.2.1 The statistical model for the individual parameters

We assume that ψ_i is a transformation of a Gaussian random vector ϕ_i:

$$\psi_i = h(\phi_i) \tag{1.3}$$

where, by rearranging the covariates (c_{im}) into a matrix C_i, ϕ_i can be written as:

$$\phi_i = C_i \mu + \eta_i \tag{1.4}$$

Examples of transformations

Here, different transformations (h_ℓ) can be used for the different components of $\psi_i = (\psi_{i,\ell})$ where $\psi_{i,\ell} = h_\ell(\phi_{i,\ell})$ for $\ell = 1, 2, \ldots, \ell$.

- $\psi_{i,\ell}$ has a normal distribution if $h_\ell(u) = u$
- $\psi_{i,\ell}$ has a log-normal distribution if $h_\ell(u) = e^u$
- assuming that $\psi_{i,\ell}$ takes its values in $(0,1)$, we can use a logit transformation $h_\ell(u) = 1/(1+e^{-u})$, or a probit transformation $h_\ell(u) = \mathbb{P}(\mathcal{N}(0,1) \leq u)$.

In the following, we will use either the parameters ψ_i or the Gaussian transformed parameters $\phi_i = h^{-1}(\psi_i)$.

The model can address continuous and/or binary covariates.

Example of continuous covariate model

Consider a PK model that depends on volume and clearance and consider the following covariate model for these two parameters:

$$\begin{aligned} CL_i &= CL_{\text{pop}} \left(\frac{W_i}{W_{\text{pop}}}\right)^{\beta_{CL,W}} \left(\frac{A_i}{A_{\text{pop}}}\right)^{\beta_{CL,A}} e^{\eta_{i,1}} \\ V_i &= V_{\text{pop}} \left(\frac{W_i}{W_{\text{pop}}}\right)^{\beta_{V,W}} e^{\eta_{i,2}} \end{aligned}$$

Where W_i and A_i are the weight and the age of subjet i and where W_{pop} and A_{pop} are some "typical" values of these two covariates in the population. Here, ψ_i will denote the PK parameters (clearance and volume) of subject i and ϕ_i its log-clearance and log-volume. Let

$$W_i^\star = \log\left(\frac{W_i}{W_{\text{pop}}}\right) \quad ; \quad A_i^\star = \log\left(\frac{A_i}{A_{\text{pop}}}\right)$$

Then,

$$\begin{aligned} \phi_i &= \begin{pmatrix} \log(CL_i) \\ \log(V_i) \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & W_i^\star & W_i^\star & 0 \\ 0 & 1 & 0 & 0 & W_i^\star \end{pmatrix} \begin{pmatrix} \log(CL_{\text{pop}}) \\ \log(V_{\text{pop}}) \\ \beta_{CL,W} \\ \beta_{CL,A} \\ \beta_{V,W} \end{pmatrix} + \begin{pmatrix} \eta_{i,1} \\ \eta_{i,2} \end{pmatrix} \\ &= C_i \mu + \eta_i \end{aligned}$$

Example of categorical covariate model

Assume that some categorical covariate G_i takes the values 1, 2, ..., K. Assume that if patient i belongs to group k, *i.e.* $G_i = k$, then

$$\log(CL_i) = \log(CL_{\text{pop},k}) + \eta_i$$

where $CL_{\text{pop},k}$ is the population clearance in group k.

Let $k^\star$ be the reference group. Then, for any group k, we will decompose the population clearance $CL_{\text{pop},k}$ as

$$\log(CL_{\text{pop},k}) = \log(CL_{\text{pop},k^\star}) + \beta_k$$

where $\beta_{k^\star} = 0$.

The variance of the random effects can also depend on this categorical covariate:

$$\eta_i \sim \mathcal{N}(0, \Omega_k) \quad \text{if} \quad G_i = k$$

Remark: It is assumed in SAEMIX 0.9 that the categorical covariate has only 2 categories (binary covariate). It is also assumed that the variance remains the same for both groups.

1.2.2 The residual error model

The within-group errors (ε_{ij}) are supposed to be Gaussian random variables with mean zero and variance 1. Furthermore, we suppose that the ε_{ij} and the η_i are mutually independent.

Different error models can be used in SAEMIX 0.9:

- the constant error model assumes that $g = a$ and $\xi = a$,
- the proportional error model assumes that $g = b\,f$ and $\xi = b$,
- a combined error model assumes that $g = a + b\,f$ and $\xi = (a, b)$,

Furthermore, all these error models can be applied to some transformation of the data:

$$t(y_{ij}) = t(f(x_{ij}, \psi_i)) + g(x_{ij}, \psi_i, \xi)\varepsilon_{ij} \tag{1.5}$$

In the current version of SAEMIX, the exponential error model is also available: it assumes that $y > 0$ and that:

$$\begin{aligned} t(y) &= \log(y) \\ y &= f e^{g\varepsilon} \end{aligned}$$

1.3 Citing SAEMIX

If you use this program in a scientific publication, we would like you to cite the following reference:

> Comets E, Lavenu A, Lavielle M. SAEMIX, an R version of the SAEM algorithm. 20th meeting of the Population Approach Group in Europe, Athens, Greece (2011), Abstr 2173. http://www.page-meeting.org/default.asp?abstract=2173

A BibTeX entry for LaTeX users is:

```
@Article{,
author ={E Comets and A Lavenu and M Lavielle},
title ={{SAEMIX}, an {R} version of the {SAEM} algorithm},
journal ={20$^{th}$ meeting of the {P}opulation {A}pproach {G}roup in {E}urope,
Athens, Greece},
type ={Poster},
note ={Abstr 2173},
url ={http://www.page-meeting.org/default.asp?abstract=2173},
year =2011 }
```

Chapter 2

Methodology and algorithms

2.1 Estimation of the parameters

2.1.1 The SAEM algorithm

We are in a classical framework of incomplete data: the observed data is $y = (y_{ij}\,; 1 \leq i \leq N\ , 1 \leq j \leq n_i)$, whereas the random parameters $(\psi = \psi_i \quad ; \quad 1 \leq i \leq N)$ are the non observed data. Then, the complete data of the model is (y, ψ). Our purpose is to compute the maximum likelihood estimator of the unknown set of parameters $\theta = (\mu, \Omega, a, b, c)$, by maximizing the likelihood of the observations $\ell(y; \theta)$.

In the case of a linear model, the estimation of the unknown parameters can be treated with the usual EM algorithm. At iteration k of EM, the E-step consists in computing the conditional expectation of the complete log-likelihood $Q_k(\theta) = \mathbb{E}\left(\log p(y, \psi; \theta) | y, \theta_{k-1}\right)$ and the M-step consists in computing the value θ_k that maximises $Q_k(\theta)$.

Following [7, 32], 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 [6] consists in replacing the usual E-step of EM by a stochastic procedure. At iteration k of SAEM:

- *Simulation-step* : draw $\psi^{(k)}$ from the conditional distribution $p(\cdot | y; \theta_k)$.

- *Stochastic approximation* : update $Q_k(\theta)$ according to

$$Q_k(\theta) = Q_{k-1}(\theta) + \gamma_k(\log p(y, \psi^{(k)}; \theta) - Q_{k-1}(\theta)) \tag{2.1}$$

  where (γ_k) is a decreasing sequence of positive numbers with $\gamma_1 = 1$.

- *Maximization-step* : update θ_k according to

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

It is shown in [6] that SAEM converges to a maximum (local or global) of the likelihood of the observations under very general conditions.

Here, the complete log-likelihood can be written

$$\begin{aligned}
\log p(y, \psi; \theta) &= \log p(y, h(\phi); \theta) \\
&= -\sum_{i,j} \log(g(x_{ij}, \psi_i, \xi)) - \frac{1}{2}\sum_{i,j}\left(\frac{y_{ij} - f(x_{ij}, \psi_i)}{g(x_{ij}, \psi_i, \xi)}\right)^2 \\
&\quad -\frac{N}{2}\log(|\Omega|) - \frac{1}{2}\sum_{i=1}^{N}(\phi_i - C_i\mu)'\Omega^{-1}(\phi_i - C_i\mu) - \frac{N_{tot} + Nd}{2}\log(2\pi)
\end{aligned}$$

where $N_{tot} = \sum_{i=1}^{N} n_i$ is the total number of observations.

First, consider a constant residual error model ($g = a$). The set of parameters to estimate is $\theta = (\mu, \Omega, a)$. Then, the complete model belongs to the exponential family and the approximation step reduces to only updating the sufficient statistics of the complete model:

$$\begin{aligned}
s_{1,i,k} &= s_{1,i,k-1} + \gamma_k \left(\phi_{i,k} - s_{1,i,k-1}\right), \quad i = 1, \ldots, N \\
s_{2,k} &= s_{2,k-1} + \gamma_k \left(\sum_{i=1}^{N} \phi_{i,k}\, \phi'_{i,k} - s_{2,k-1}\right) \\
s_{3,k} &= s_{3,k-1} + \gamma_k \left(\sum_{i,j}\left(y_{ij} - f(x_{ij}, \psi_i^{(k)})\right)^2 - s_{3,k-1}\right).
\end{aligned}$$

Then, θ_{k+1} is obtained in the maximization step as follows:

$$\mu_{k+1} = \left(\sum_{i=1}^{N} C_i{}'\Omega_k{}^{-1}C_i\right)^{-1} \sum_{i=1}^{N} C_i{}'\Omega_k{}^{-1}s_{1,i,k} \tag{2.2}$$

$$\Omega_{k+1} = \frac{1}{N}\left(s_{2,k} - \sum_{i=1}^{N}(C_i\mu_{k+1})s'_{1,i,k} - \sum_{i=1}^{N} s_{1,i,k}(C_i\mu_{k+1})' + \sum_{i=1}^{N}(C_i\mu_{k+1})(C_i\mu_{k+1})'\right) \tag{2.3}$$

$$a_{k+1} = \sqrt{\frac{s_{3,k}}{N_{tot}}} \tag{2.4}$$

Remark 1: The sequence of step sizes used in SAEMIX decreases as k^{-a}. More precisely, for any sequence of integers $K_1, K_2, \ldots, K_J$ and any sequence $a_1, a_2, \ldots, a_J$ of real numbers such that $0 \leq a_1 < a_2 < \ldots < a_J \leq 1$, we define the sequence of step sizes (γ_k) as follows:

$$\gamma_k = \frac{1}{k^{a_1}} \quad \text{for any } 1 \leq k \leq K_1 \tag{2.5}$$

and for $2 \leq j \leq J$,

$$\gamma_k = \frac{1}{\left(k - K_{j-1} + \gamma_{K_{j-1}}^{-1/a_j}\right)^{a_j}} \quad \text{for any } \sum_{i=1}^{j-1} K_i + 1 \leq k \leq \sum_{i=1}^{j} K_i \tag{2.6}$$

Here, $K = \sum_{j=1}^{J} K_j$ is the total number of iterations.

We recommend to use $a_1 = 0$ (that is $\gamma_k = 1$) during the first iterations, and $a_J = 1$ during the last iterations. Indeed, the initial guess θ_0 may be far from the maximum likelihood value we are looking for and the first iterations with $\gamma_k = 1$ allow to converge quickly to a neighborhood of the maximum likelihood estimator. Then, smaller step sizes ensure the almost sure convergence of the algorithm to the maximum likelihood estimator.

In the case where $J = 2$ with $a_1 = 0$ and $a_2 = 1$, the sequence of step sizes is

$$\begin{aligned} \gamma_k &= 1 && \text{for } 1 \leq k \leq K_1 \\ &= \frac{1}{k - K_1 + 1} && \text{for } K_1 + 1 \leq k \leq K_1 + K_2 \end{aligned}$$

Remark 2: The estimated covariance matrix Ω_{k+1} defined in (2.3) is a full covariance matrix. However, the covariance matrix Ω of the random effects can have any covariance structure. If we assume, for example, that there is no correlation between the random effects, we will set to 0 the non diagonal elements of Ω_{k+1} defined in (2.3).

We can also assume that a random effect has no variance. If the ℓth random effect has a variance equal to 0, then the ℓth individual parameter is no longer random and the simulation step of SAEM needs some modification. During the first K_0 iterations, we use SAEM as it was described above, considering that all the effects are random and assuming that there is no correlation between the ℓth random effect and the other ones ($\omega^2_{\ell\ell'} = 0$ for any $\ell \neq \ell'$). Then, during the next iterations, we use again SAEM, but the variance of this random effect is no longer estimated: it is forced to decrease at each iteration by setting

$$\omega^2_{\ell\ell,k+1} = \alpha\, \omega^2_{\ell\ell,k} \quad , \quad K_0 \leq k \leq K \tag{2.7}$$

where α is chosen between 0 and 1 such that $\omega^2_{\ell\ell,K} = 10^{-6} \omega^2_{\ell\ell,K_0}$.

Remark 3: - For a residual variance model of the form $g = b\, f^c$, where c is fixed, the complete model also belongs to the exponential family and the estimation of b is straightforward: the sufficient statistics sequence $(s_{3,k})$ is defined by

$$s_{3,k} = s_{3,k-1} + \gamma_k \left(\sum_{i,j} \left(\frac{y_{ij} - f(x_{ij}, \psi_i^{(k)})}{f^c(x_{ij}, \psi_i^{(k)})} \right)^2 - s_{3,k-1} \right)$$

and $b_{k+1} = \sqrt{s_{3,k}/N_{tot}}$.

- For a general residual variance model $g = a + b\, f^c$, the complete model does not belong to the exponential family and the estimates of the residual variance parameters (a, b, c) cannot be expressed as a function of some sufficient statistics. Then, let (A_k, B_k, C_k) that minimise the complete log-likelihood:

$$(A_k, B_k, C_k) = \text{Arg} \min_{(a,b,c)} \left\{ \sum_{i,j} \log(a + bf^c(x_{ij}, \psi_i^{(k)})) + \frac{1}{2} \sum_{i,j} \left(\frac{y_{ij} - f(x_{ij}, \psi_i^{(k)})}{a + bf^c(x_{ij}, \psi_i^{(k)})} \right)^2 \right\}$$

We update the residual variance parameters as follows:

$$a_{k+1} = a_k + \gamma_k\,(A_k - a_k) \quad (2.8)$$
$$b_{k+1} = b_k + \gamma_k\,(B_k - b_k) \quad (2.9)$$
$$c_{k+1} = c_k + \gamma_k\,(C_k - c_k) \quad (2.10)$$

The estimation of μ and Ω remains unchanged.

2.1.2 The MCMC-SAEM algorithm

For model (1.1), the simulation step cannot be directly performed. Kuhn and Lavielle [12] propose to combine the SAEM algorithm with a MCMC (Markov Chain Monte Carlo) procedure. This procedure consists in replacing the Simulation-step at iteration k by m iterations of the Hastings-Metropolis algorithm.

Here, we will consider the Gaussian parameters (ϕ_i). For $i = 1, 2, \ldots, N$

- let $\phi_{i,0} = \phi_i^{(k-1)}$
- for $p = 1, 2, \ldots, m$,
  1. draw $\tilde{\phi}_{i,p}$ using the proposal kernel $q_{\theta_k}(\phi_{i,p-1}, \cdot)$
  2. set $\phi_{i,p} = \tilde{\phi}_{i,p}$ with probability

  $$\alpha(\phi_{i,p-1}, \tilde{\phi}_{i,p}) = \min \left(1, \frac{p(\tilde{\phi}_{i,p}|y_i; \theta_k) q_{\theta_k}(\tilde{\phi}_{i,p}, \phi_{i,p-1})}{p(\phi_{i,p-1}|y_i; \theta_k) q_{\theta_k}(\phi_{i,p-1}, \tilde{\phi}_{i,p})} \right)$$

  and $\phi_{i,p} = \phi_{i,p-1}$ with probability $1 - \alpha(\phi_{i,p-1}, \tilde{\phi}_{i,p})$.

- let $\phi_i^{(k)} = \phi_{i,m}$.

Several transition kernels, associated to different proposals can be successively used. We use the four following proposal kernels:

1. $q_{\theta_k}^{(1)}$ is the prior distribution of ϕ_i at iteration k, that is the Gaussian distribution $\mathcal{N}(C_i\mu_k, \Omega_k)$ and then

$$\alpha(\phi_{i,p-1}, \tilde{\phi}_{i,p}) = \min\left(1, \frac{p(y_i|\tilde{\phi}_{i,p};\theta_k)}{p(y_i|\phi_{i,p-1};\theta_k)}\right)$$

2. $q_{\theta_k}^{(2)}$ is a random permutation of the ϕ_i: generate a random permutation σ of $\{1, 2, \ldots, N\}$ and set $\tilde{\phi}_{i,p} = \phi_{\sigma(i),p-1}$.

3. $q_{\theta_k}^{(3)}$ is the multidimensional random walk $\mathcal{N}(\phi_{i,p-1}, \kappa\Omega_k)$. This kernel is symmetric and then

$$\alpha(\phi_{i,p-1}, \tilde{\phi}_{i,p}) = \min\left(1, \frac{p(y_i, \tilde{\phi}_{i,p};\theta_k)}{p(y_i, \phi_{i,p-1};\theta_k)}\right)$$

4. $q_{\theta_k}^{(4)}$ is a succession of d unidimensional Gaussian random walks: each component of ϕ_i are successively updated.

Then, the simulation-step at iteration k consists in running m_1 iterations of the Hasting-Metropolis with proposal $q_{\theta_k}^{(1)}$, m_2 iterations with proposal $q_{\theta_k}^{(2)}$, m_3 iterations with proposal $q_{\theta_k}^{(3)}$ and m_4 iterations with proposal $q_{\theta_k}^{(4)}$.

Remark 1 : During the first K_b iterations ("burning" iterations) of SAEM, we only run the MCMC algorithm but the parameters are not updated.

Remark 2 : When the number N of subjects is small, convergence of the algorithm can be improved by running L Markov Chain instead of only one. The simulation step requires to draw L sequences $\phi^{(k,1)}, \ldots, \phi^{(k,L)}$ at iteration k and to combine stochastic approximation and Monte Carlo in the approximation step:

$$Q_k(\theta) = Q_{k-1}(\theta) + \gamma_k\left(\frac{1}{L}\sum_{\ell=1}^{L}\log p(y, \phi^{(k,\ell)};\theta) - Q_{k-1}(\theta)\right) \tag{2.11}$$

2.1.3 The Simulated Annealing SAEM algorithm

Convergence of SAEM can strongly depend on the initial guess if the likelihood ℓ possesses several local maxima. The Simulated Annealing version of SAEM improves the convergence of the algorithm toward the global maximum of ℓ.

For the sake of simplicity, we will consider here a constant residual error model $g = a$. Let

$$U(y, \phi; \theta) = \frac{1}{2a^2} \sum_{i,j} (y_{ij} - f(x_{ij}, h(\phi_i)))^2 + \frac{1}{2} \sum_{i=1}^{N} (\phi_i - C_i\mu)'\Omega^{-1}(\phi_i - C_i\mu)$$

Then, we can write the complete likelihood:

$$p(y, \phi; \theta) = C(\theta)\, e^{-U(y,\phi;\theta)}$$

where $C(\theta)$ is a normalizing constant that only depends on θ.

For any *temperature* $T \geq 0$, we consider the complete model

$$p_T(y, \phi; \theta) = C_T(\theta)\, e^{-\frac{1}{T}U(y,\phi;\theta)}$$

where $C_T(\theta)$ is a normalizing constant. This model consists in replacing the variance matrix Ω by $T\Omega$ and the residual variance a^2 by Ta^2. In other words, a model "with a large temperature" is a model with large variances.

We introduce a decreasing temperature sequence $(T_k, 1 \leq k \leq K)$ and use the MCMC-SAEM algorithm considering the complete model $p_{T_k}(y, \phi; \theta)$ at iteration k (while the usual version of MCMC-SAEM uses $T_k = 1$ at each iteration). The sequence (T_k) is large during the first iterations and decreases to 1 with exponential rate. This is done by choosing large initial variances Ω_0 and a_0^2 and setting

$$\tilde{\Omega}_{k+1} = \frac{1}{N}\left(s_{2,k} - \sum_{i=1}^{N}(C_i\mu_{k+1})s'_{1,i,k} - \sum_{i=1}^{N} s_{1,i,k}(C_i\mu_{k+1})' + \sum_{i=1}^{N}(C_i\mu_{k+1})(C_i\mu_{k+1})'\right) \quad (2.12)$$

$$a_{k+1} = \sqrt{\frac{s_{3,k}}{N_{tot}}} \quad (2.13)$$

$$\Omega_{k+1} = \max\left(\tau\Omega_k \;,\; \tilde{\Omega}_{k+1}\right) \quad (2.14)$$

$$a_{k+1}^2 = \max\left(\tau a_k^2 \;,\; \frac{s_{3,k}}{N}\right) \quad (2.15)$$

during the first iterations of the algorithm and where $0 \leq \tau \leq 1$.

These large values of the variances make the conditional distribution $p(\phi|y;\theta)$ less concentrated around its mode. This procedure allows the sequence (θ_k) to escape from the local maxima of the likelihood and to converge to a neighborhood of the global maximum of ℓ. After that, the usual MCMC-SAEM algorithm is used, estimating the variances at each iteration.

Remark 1: The Simulated Annealing version of SAEM is performed during the first K_{sa} iterations. Of course, SAEM without any simulated annealing can be run by setting $\tau = 0$. On the other hand, simulated annealing is obtained with τ close to 1.

Remark 2: We can use two different coefficients τ_1 and τ_2 for Ω and a^2 in SAEMIX. It is possible, for example, to choose $\tau_1 < 1$ and $\tau_2 > 1$, with a small initial residual variance and large initial inter-subject variances. In this case, SAEM tries to obtain the best possible fit during the first iterations, allowing a large inter-subject variability. During the next iterations, this variability is reduced and the residual variance increases until reaching the best possible trade-off between these two criteria.

2.2 Estimation of the Fisher Information matrix

Let $\theta^\star$ be the true unknown value of θ, and let $\widehat{\theta}$ be the maximum likelihood estimate of θ. If the observed likelihood function ℓ is sufficiently smooth, asymptotic theory for maximum-likelihood estimation holds and

$$\sqrt{N}(\widehat{\theta} - \theta^\star) \underset{\mathrm{N}\to\infty}{\longrightarrow} \mathcal{N}(0, I(\theta^\star)^{-1}) \tag{2.16}$$

where $I(\theta^\star) = -\partial_\theta^2 \log \ell(y;\theta^\star)$ is the true Fisher information matrix. Thus, an estimate of the asymptotic covariance of $\widehat{\theta}$ is the inverse of the Fisher information matrix $I(\widehat{\theta}) = -\partial_\theta^2 \log \ell(y;\widehat{\theta})$.

2.2.1 Linearization of the model

The Fisher information matrix of the nonlinear mixed effects model defined in (1) cannot be computed in a closed-form.

An alternative is to approximate this information matrix by the Fisher information matrix of the Gaussian model deduced from the nonlinear mixed effects model after linearization of the function f around the conditional expectation of the individual Gaussian parameters $(\mathbb{E}\left(\phi_i|y;\hat{\theta}\right), 1 \le i \le N)$. The Fisher information matrix of this Gaussian model is a block matrix (no correlations between the estimated fixed effects and the estimated variances). The gradient of f is numerically computed.

Remark 1: We do not recommend the linearization of the model to estimate the parameters of the model, as it is done with the FO and FOCE algorithms. On the other hand, many numerical experiments have shown that this approach can be used to estimate the Fisher information matrix.

Remark 2: Obviously, this approach cannot be used with discrete data models...

2.2.2 A stochastic approximation of the Fisher Information Matrix

It is possible to obtain an estimation of the Fisher information matrix using the Louis's missing information principle [18]:

$$\partial_\theta^2 \log \ell(y;\theta) = \mathrm{E}\big(\partial_\theta^2 \log p(y,\phi;\theta)|y;\theta\big) + \mathrm{Cov}\big(\partial_\theta \log p(y,\phi;\theta)|y;\theta\big) \tag{2.17}$$

where

$$\begin{aligned}\mathrm{Cov}\big(\partial_\theta \log p(y,\phi;\theta)|y;\theta\big) &= \mathrm{E}\big(\partial_\theta \log p(y,\phi;\theta)\partial_\theta \log p(y,\phi;\theta)'|y;\theta\big) \\ &\quad - \mathrm{E}\big(\partial_\theta \log p(y,\phi;\theta)|y;\theta\big)\mathrm{E}\big(\partial_\theta \log p(y,\phi;\theta)|y;\theta\big)'\end{aligned}$$

and

$$\partial_\theta \log g(y;\theta) = \mathrm{E}\big(\partial_\theta \log p(y,\phi;\theta)|y;\theta\big)$$

Here, $\partial_\theta u$ is the gradient of u (*i.e.* the vector of first derivatives of u with respect to θ) and $\partial_\theta^2 u$ is the hessian of u (*i.e.* the matrix of second derivatives of u with respect to θ).

Then, using SAEM, the matrix $\partial_\theta^2 \log \ell(y;\widehat{\theta})$ can be approximated by the sequence (H_k) defined as follows:

$$\begin{aligned}\Delta_k &= \Delta_{k-1} + \gamma_k \left(\partial_\theta \log f(y,\phi_k;\theta_k) - \Delta_{k-1}\right) \\ D_k &= D_{k-1} + \gamma_k \left(\partial_\theta^2 \log f(y,\phi_k;\theta_k) - D_{k-1}\right) \\ G_k &= G_{k-1} + \gamma_k \left(\partial_\theta \log f(y,\phi_k;\theta_k)\partial_\theta \log f(y,\phi_k;\theta_k)^t - G_{k-1}\right) \\ H_k &= D_k + G_k - \Delta_k\Delta_k^t\end{aligned}$$

In the current version of SAEMIX, only the linearisation approach has been implemented.

2.3 Estimation of the individual parameters

When the parameters of the model have been estimated, we can estimate the individual parameters (ψ_i). To do that, we will estimate the individual normally distributed parameters (ϕ_i) and derive the estimates of (ψ_i) using the transformation $\psi_i = h(\psi_i)$.

Let $\hat{\theta}$ be the estimated value of θ computed with the SAEM algorithm and let $p(\phi_i|y_i;\hat{\theta})$ be the conditional distribution of ϕ_i for $1 \leq i \leq N$.

We use the MCMC procedure used in the SAEM algorithm to estimate these conditional distributions. More precisely, for $1 \leq i \leq N$, we empirically estimate:

- the conditional mode (or Maximum A Posteriori) $m(\phi_i|y_i;\hat{\theta}) = \mathrm{Arg\,max}_{\phi_i}\, p(\phi_i|y_i;\hat{\theta})$,
- the conditional mean $E(\phi_i|y_i;\hat{\theta})$,
- the conditional standard deviation $sd(\phi_i|y_i;\hat{\theta})$.

Remarks:

1. The prior distribution of ϕ_i is a normal distribution, but not the conditional distribution $p(\phi_i|y_i;\hat{\theta})$ (remember that the structural model is not a linear function of $\phi_i\ldots$). Then, the conditional mode $m(\phi_i|y_i;\hat{\theta})$ and the conditional expectation $E(\phi_i|y_i;\hat{\theta})$ are two different predictors of ϕ_i.
2. If the transformation h is not linear,

$$\begin{aligned} \mathbb{E}\left(\psi_i|y_i;\hat{\theta}\right) &= \mathbb{E}\left(h(\phi_i|y_i;\hat{\theta}\right) \\ &\neq h\left(\mathbb{E}\left(\phi_i|y_i;\hat{\theta}\right)\right) \end{aligned}$$

   In SAEMIX, we estimate $\mathbb{E}\left(\phi_i|y_i;\hat{\theta}\right)$ and $\mathbb{E}\left(\psi_i|y_i;\hat{\theta}\right)$.

The number of iterations of the MCMC algorithm used to estimate the conditional mean and standard deviation is adaptively chosen as follows:

1. the (ϕ_i) are initialised with the last value obtained in SAEM
2. we run the Hastings-Metropolis with kernel $q^{(1)}$, $q^{(3)}$ and $q^{(4)}$ and compute at each iteration the empirical conditional mean and s.d. of ϕ_i:

$$e_{i,K} = \frac{1}{K}\sum_{k=1}^{K}\phi_{i,k} \tag{2.18}$$

$$sd_{i,K} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\phi_{i,k}^2 - e_{i,K}^2} \tag{2.19}$$

   where $\phi_{i,k}$ is the value of ϕ_i at iteration k of the MCMC algorithm.

3. we stop the algorithm at iteration K and use $e_{i,K}$ and $sd_{i,K}$ to estimate the conditional mean and s.d. of ϕ_i if, for any $K - L_{mcmc} + 1 \leq k \leq K$,

$$\begin{aligned}(1-\rho_{mcmc})\bar{e}_K &\leq \bar{e}_k \leq (1+\rho_{mcmc})\bar{e}_K \\ (1-\rho_{mcmc})\bar{sd}_K &\leq \bar{sd}_k \leq (1+\rho_{mcmc})\bar{sd}_K\end{aligned} \qquad (2.20)$$

   where $0 < \rho_{mcmc} < 1$. That means that the sequence of empirical means and s.d. must stay in a ρ_{mcmc}-confidence interval during L_{mcmc} iterations.

2.4 Estimation of the likelihood

2.4.1 Linearization of the model

The likelihood of the nonlinear mixed effects model defined in (1) cannot be computed in a closed-form.

An alternative is to approximate this likelihood 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 $(\phi_i, 1 \leq i \leq N)$.

2.4.2 Estimation using importance sampling

The likelihood of the observations can be estimated without any approximation using a Monte-Carlo approach. The likelihood ℓ of the observations can be decomposed as follows:

$$\begin{aligned}\ell(y;\theta) &= \int p(y,\phi;\theta)\,d\phi \\ &= \int h(y|\phi;\theta)\pi(\phi;\theta)\,d\phi\end{aligned}$$

where π is the so-called *prior distribution* of ϕ. According to (1.2), π is a Gaussian distribution.

For any distribution $\tilde{\pi}$ absolutely continuous with respect to the prior distribution π, we can write

$$\ell(y;\theta) = \int h(y|\phi;\theta)\frac{\pi(\phi;\theta)}{\tilde{\pi}(\phi;\theta)}\tilde{\pi}(\phi;\theta)\,d\phi$$

Then, $\ell(y;\theta)$ can be approximated via an *Importance Sampling* integration method:

1. draw $\phi^{(1)}, \phi^{(2)}, \ldots, \phi^{(M)}$ with the distribution $\tilde{\pi}(\cdot;\theta)$,

2. let

$$\ell_M(y;\theta) = \frac{1}{M}\sum_{j=1}^{M} h(y|\phi^{(j)};\theta)\frac{\pi(\phi^{(j)};\theta)}{\tilde{\pi}(\phi^{(j)};\theta)} \tag{2.21}$$

The statistical properties of the estimator $\ell_M(y;\theta)$ of the likelihood $\ell(y;\theta)$ strongly depend on the sampling distribution $\tilde{\pi}$. First, note that

$$\begin{aligned} \mathbb{E}\left(\ell_M(y;\theta)\right) &= \ell(y;\theta), \\ \mathrm{Var}\left(\ell_M(y;\theta)\right) &= \mathcal{O}(1/M). \end{aligned}$$

Furthermore, if $\tilde{\pi}$ is the conditional distribution $p(\phi|y;\theta)$, the variance of the estimator is null and $\hat{\ell}_M(y;\theta) = \ell(y;\theta)$ for any value of M. That means that an accurate estimation of $\ell(y;\theta)$ can be obtained with a small value of M if the sampling distribution is close to the conditional distribution $p(\phi|y;\theta)$.

In SAEMIX, for $i = 1, 2, \ldots, N$, we empirically estimate the conditional mean $\mathbb{E}\left(\phi_i|y_i;\widehat{\theta}\right)$ and the conditional variance $\mathrm{Var}\left(\phi_i|y_i;\widehat{\theta}\right)$ of ϕ_i as described above. Then, the ${\phi_i}^{(j)}$ are drawn with the sampling distribution $\tilde{\pi}$ as follows:

$${\phi_i}^{(j)} = \mathbb{E}\left(\phi_i|y_i;\widehat{\theta}\right) + \mathrm{Var}\left(\phi_i|y_i;\widehat{\theta}\right)^{\frac{1}{2}} \times T_{ij}$$

where (T_{ij}) is a sequence of *i.i.d.* random variables distributed with a $t-$distribution with ν degrees of freedom. In the current version of SAEMIX, the default value is $\nu = 5$.

The quality of the approximation depends on the estimates of the conditional mean and variances of the individual distributions.

2.4.3 Estimation using Gaussian Quadrature

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 $\mathbb{E}\left(\phi_i|y_i;\widehat{\theta}\right)$ and $\mathrm{Var}\left(\phi_i|y_i;\widehat{\theta}\right)$.

2.5 Model predictions

2.5.1 Population predictions

Population predictions represent the predictions from the model in the absence of data, and they only take into account individual design variables (eg dose regimen) and covariates.

Two types of population predictions are available in SAEMIX:

1. the predictions using the population parameters: $f\left(x_{ij}; h\left(\mathbb{E}_{\hat{\theta}}(\phi_i)\right)\right) = f(x_{ij}; h(C_i\hat{\mu}))$. These are provided in the output under the name `ypred`

2. the population mean predictions: $\mathbb{E}_{\hat{\theta}}(f(x_{ij}; \psi_i)) = \mathbb{E}_{\hat{\theta}}(f(x_{ij}; h(\phi_i)))$. These are provided in the output under the name `ppred`

2.5.2 Individual predictions

Individual predictions take into account not only covariates and individual design variables such as dose regimen, but also use the observations in that individual to obtain the parameters providing the best fit for that particular subject, given the population parameters.

In section 2.3, we described how the conditional distribution of the parameters for each individual is obtained in SAEMIX. Two types of individual parameters are reported in the output:

1. the conditional mode (or Maximum A Posteriori): $m(\phi_i|y_i;\hat{\theta}) = \text{Arg}\max_{\phi_i} p(\phi_i|y_i;\hat{\theta})$. These are reported in the output as `map.psi`

2. the conditional mean: $E(\phi_i|y_i;\hat{\theta})$. These are reported in the output as `cond.mean.psi`

Correspondingly, two types of individual predictions can be obtained in SAEMIX:

1. the predictions obtained using the conditional mode are reported in the output as `ipred`

2. the predictions obtained using the conditional mean are reported in the output as `icpred`

2.6 Estimation of the weighted residuals

2.6.1 Population Weighted Residuals

The vector of Population Weighted Residuals are evaluated as:

$$PWRES_i = Var_{\hat{\theta}}(y_i)^{-1/2}\,(y_i - \hat{y}_i^{pop})$$

where $\hat{y}_{ij}^{pop}$ is the population prediction of y_{ij} and $Var_{\hat{\theta}}(y_{ij})$ is the variance-covariance matrix of y_i.

Weighted residuals are computed using the population mean predictions `ppred` $\mathbb{E}_{\hat{\theta}}(f(x_{ij};\psi_i))) = \mathbb{E}_{\hat{\theta}}(f(x_{ij};h(\phi_i)))$ for $\hat{y}_{ij}^{pop}$. $\mathbb{E}_{\hat{\theta}}(f(x_{ij};h(\phi_i))$ and $Var_{\hat{\theta}}(y_{ij})$ are estimated with a Monte-Carlo procedure.

Remark: This computation is performed during the computation of pd/npde, so that a basic SAEMIX object does not include these elements.

2.6.2 Individual Weighted Residuals

The Individual Weighted Residuals are evaluated as

$$IWRES_{ij} = \frac{y_{ij} - \hat{y}_{ij}^{ind}}{\hat{\sigma}_{ij}^{ind}}$$

where $\hat{y}_{ij}^{ind} = f(x_{ij};\hat{\psi}_i)$ is the individual prediction of y_{ij} and $(\hat{\sigma}_{ij}^{ind})^2 = g(x_{ij};\hat{\phi}_i,\hat{\xi})^2$ is the residual variance of y_{ij}.

The two types of individual parameters described in section 2.5 yield two types of individual weighted residuals in the SAEMIX output:

1. the individual weighted residuals obtained using the conditional mode are reported in the output as `iwres`

2. the individual weighted residuals obtained using the conditional mean are reported in the output as `icwres`

Remark: When a transformed residual error model is used (an exponential error model for instance), the weighted residuals are computed using $t(y)$ instead of y.

2.6.3 Normalised Prediction Distribution Errors

The Normalised Prediction Distribution Errors are defined as follow

$$\text{npde}_{ij} = \Phi^{-1}(\hat{p}_{ij})$$

where Φ is the $\mathcal{N}(0,1)$ cumulative distribution function and where $\hat{p}_{ij}$ is an empirical estimator of

$$p_{ij} = \mathbb{P}(Y_{ij} < y_{ij})$$

obtained by Monte-Carlo.

In more details, prediction discrepancies (pd) are first obtained, as the percentile of the observation in the cumulative distribution function F_{ij} of the predictive distribution of Y_{ij} under the model being evaluated. F_{ij} is obtained by simulating K datasets under the model, and the corresponding prediction discrepancies is given by:

$$\text{pd}_{ij} = F_{ij}(y_{ij}) \approx \frac{1}{K}\sum_{k=1}^{K}\delta_{ijk} \tag{2.22}$$

where $\delta_{ijk} = 1$ if $y_{ij}^{sim(k)} < y_{ij}$ and 0 otherwise, $y_{ij}^{sim(k)}$ denoting the value of y_{ij} simulated in the k^{th} replication.

To handle correlations within the observations obtained in the same individual, we first compute the empirical mean $\hat{\mathbf{E}}(\mathbf{y}_i)$ and empirical variance-covariance matrix Var $(()\,\mathbf{y}_i)$ over the K simulations. Decorrelation is performed simultaneously for simulated data:

$$\mathbf{y}_i^{sim(k)*} = \hat{\mathbf{V}}_i^{-1/2}(\mathbf{y}_i^{sim(k)} - \hat{\mathbf{E}}(\mathbf{y}_i)) \tag{2.23}$$

and for observed data:

$$\mathbf{y}_i^{*} = \hat{\mathbf{V}}_i^{-1/2}(\mathbf{y}_i - \hat{\mathbf{E}}(\mathbf{y}_i)) \tag{2.24}$$

Decorrelated pd are then obtained using the same formula as in (2.22) but with the decorrelated data, and we call the resulting variables prediction distribution errors (pde):

$$\text{pde}_{ij} = F_{ij}^{*}(y_{ij}^{*}) \approx \frac{1}{K}\sum_{k=1}^{K}\delta_{ijk}^{*} \tag{2.25}$$

where $\delta_{ijk}^{*} = 1$ if $y_{ij}^{sim(k)*} < y_{ij}^{*}$ and 0 otherwise.

Normalised prediction distribution errors (npde) are then obtained as:

$$\text{npde}_{ij} = \Phi^{-1}(\text{pde}_{ij}) \tag{2.26}$$

Remark: the empirical mean and covariance-matrix computed here are also used for the decorrelation step in the computation of the population weighted residuals, $WRES$. The $WRES$ in SAEMIX are thus computed in conjunction with the more advanced metric npde.

2.7 Inputs and outputs

2.7.1 The inputs

To summarise, SAEMIX requires to define the model and to fix some parameters used for the algorithms. First, it is necessary to define:

- the structural model, that is the regression function f defined in (1.1),
- the covariate model, that is the structure of the matrix μ defined in (1.2) and the covariates (c_i).
- the variance-covariance model for the random effects, that is the structure of the variance-covariance matrix Ω defined in (1.2).
- the residual variance model, that is the regression function g.

The only mandatory elements for a SAEMIX fit are:

- a data object, defined by at least:
  - the name of the data file
  - we advise to also specify the names of the columns containing the grouping variable, the predictor(s) and the response, although the program will attempt to recognise suitable columns
- a model object, defined by at least:
  - the name of a valid model function
  - the matrix of starting values `psi0`
    - if no covariates are present in the model, a single line is sufficient, which will contain the starting values for the fixed effects μ in the model
    - if covariates are present in the model: if `psi0` has more than 1 line, the next lines are assumed to represent the starting values for the covariate models (only parameters actually present in the model will be estimated, even if `psi0` contains non-null values; otherwise, values of 0 will be assumed.

Then, it is necessary to specify several parameters for running the algorithms:

- the SAEM algorithm requires to specify
  - the initial values of the fixed effects μ_0, the initial variance covariance matrix Ω_0 of the random effects and the initial residual variance coefficients a_0, b_0 and c_0,
  - the sequence of step sizes (γ_k), that is the numbers of iterations (K_1, K_2) and the coefficients (a_1, a_2) defined in (2.5) and (2.6),
  - the number of burning iterations K_b used with the same value θ_0 before updating the sequence (θ_k).
- the MCMC algorithm requires to set
  - the number of Markov Chains L,
  - the numbers m_1, m_2, m_3 and m_4 of iterations of the Hasting-Metropolis algorithm,
  - the probability of acceptance ρ for kernel $q^{(3)}$ and $q^{(4)}$,
- the algorithm to estimate the conditional distribution of the (ϕ_i) requires to set
  - the width of the confidence interval ρ_{mcmc} (see (2.20),
  - the number of iterations L_{mcmc}.
- the Simulated Annealing algorithm requires to set
  - the coefficient τ_1 and τ_2 defining the decrease of the temperature (see (2.14,2.15))
  - the number of iterations K_{sa}.
- the Importance Sampling algorithm requires to set
  - the Monte Carlo number M used to estimate the observed likelihood (see (2.21)).
- the Gaussian Quadrature algorithm requires to set
  - the number of quadrature points N_{QG} used to compute each integral (see (2.4.3))
  - the width of each integral N_{QG}

In the R implementation of SAEMIX, most of these parameters, as well as other variables used by the algorithm, are set through a list which is included in the object returned by an SAEMIX fit. Table 2.1 shows the correspondance between the parameters and the elements in this list.

Parameter	Meaning	Option name	Default value
L	number of Markov Chains	nb.chains	1*
K_1, K_2	Number of iterations during the two periods	nbiter.saemix	c(300,100)
K_b	Number of burning iterations	nbiter.burn	5
m_1, m_2, m_3	Number of iterations of kernels $q^{(2)}$, $q^{(3)}$ and $q^{(4)}$ at each iteration of SAEM	nbiter.mcmc	c(2,2,2)
	Number of iterations during which simulated annealing is performed	nbiter.sa	
ρ	Probability of acceptance for kernels $q^{(2)}$ and $q^{(3)}$	proba.mcmc	0.4
	Stepsize for kernels $q^{(2)}$ and $q^{(3)}$	stepsize.rw	0.4
	Initial variance parameters for kernels $q^{(2)}$ and $q^{(3)}$	rw.init	0.5
τ	Parameter controlling cooling in the Simulated Annealing algorithm	alpha.sa	0.97
M	Number of Monte-Carlo samples used to estimate the likelihood by Importance Sampling	nmc.is	5000
ν	Number of degrees of freedom of the Student distribution used for the estimation of the log-likelihood by Importance Sampling	nu.is	4
K_{GQ}	Number of nodes used for Gaussian Quadrature	nnodes.gq	12
	Width of the distribution used for Gaussian Quadrature (in SD)	nsd.gq	4
L_{mcmc}	Number of iterations required to assume convergence for the conditional estimates	ipar.lmcmc	50
ρ_{mcmc}	Confidence interval for the conditional mean and variance	ipar.rmcmc	0.95
Other variables			
	Algorithms to be run in a call to saemix(): a vector of 3 values of 0/1, representing respectively individual parameter estimates (MAP), estimation of the Fisher information matrix and estimation of the LL by importance sampling	algorithms	c(1,1,1)
	Plot graphs during the estimation of the LL by IS	print.is	FALSE
	Maximum number of iterations for the estimation of fixed effects	maxim.maxiter	100
	Whether convergence plots should be drawn at regular intervals during the estimation	displayProgress	TRUE

– To be continued

Table 2.1 – cont.

Parameter	Meaning	Option name	Default value
	Interval (in number of iterations) between two convergence plots	nbdisplay	
	Seed to initialise the random number generator	seed	123456

Table 2.1: Parameters set as options in the options list. To set an option, one would define it as an element of this list (see examples), and any option not defined by the user is automatically set to its default value.

* *the default number of chains is 1, except when the number of subjects is smaller than 50, where it defaults to n_c where n_c is the smallest integer such that $n_c\ N \geq 50$*

Assuming the result of the SAEMIX fit has been stored in an object saemix.fit, the list of options can be accessed using the following instruction (see section 4.2 for more details on how to access elements of objects in R):

```
saemix.fit["options"]
```

For example, to see the number of chains, one would type in R:

```
saemix.fit["options"]$nb.chains
```

The easiest way to set options is to pass them in a list when calling the main fitting function, as can be seen in the example section (section 3.1).

2.7.2 The outputs

In the R implementation of SAEMIX, the object returned after a call to the main fitting function saemix() contains the following elements:

- data: the data object, created by a call to the saemixData() function, and containing the dataset to be used in the analysis
- model: the model object, created by a call to the saemixModel() function, and containing the model characteristics
- options: a list containing the options for the estimation algorithm (see above)
- prefs: a list containing the graphical preferences for plots, which will be described in the next section
- results: the results object
- rep.data: the replicated data (when available)
- sim.data: the simulated data (when available)

Assuming the result of a call to saemix() has been ascribed to the object `yfit`, these elements can be accessed, for example for the results element, with the following command:

```
yfit["results"]
```

The results object is an object of class SaemixRes. Most users will not need to access the elements since functions have been created to output the results. However, elements of the results object can also be accessed individually; for example, the likelihood estimated by importance sampling can be accessed as:

```
yfit["results"]["ll.is"]
```

More details on S4 structures (objects and methods), and on how to access the elements of S4 objects can be found in 4.

Table 2.2 shows the most important elements present in the results object (some of these are only present after a call to a specific function, or when the proper option has been set; for instance, estimates of individual parameters are only estimated when the first element of the algorithm element in options is 1).

Element	Meaning
npar.est	Number of parameter estimates
fixed.effects	Estimates of the fixed effects
se.fixed	Standard errors of estimation of the fixed effects
respar	Estimates of the parameters of the residual error model
se.repar	Standard errors of estimation of the residual parameters
omega	Estimates of the fixed effects
se.omega	Standard errors of the estimation of the fixed effects
ll.is	Log-likelihood estimated by importance sampling
aic.is	AIC using the log-likelihood estimated by importance sampling
bic.is	BIC using the log-likelihood estimated by importance sampling
ll.lin	Log-likelihood estimated by linearisation
aic.lin	AIC using the log-likelihood estimated by linearisation
bic.lin	BIC using the log-likelihood estimated by linearisation
ll.gq	Log-likelihood estimated by gaussian quadrature
aic.gq	AIC using the log-likelihood estimated by gaussian quadrature
bic.gq	BIC using the log-likelihood estimated by gaussian quadrature
map.psi	Individual estimates of the parameters (ψ), obtained as the mode of the conditional distribution (MAP)
map.phi	Estimate of the corresponding individual ϕ
map.eta	Estimate of the corresponding random effect
map.shrinkage	Shrinkage for the MAP estimates
cond.mean.psi	Individual estimates of the parameters, obtained as the mean of the conditional distribution
cond.mean.phi	Estimate of the corresponding individual ϕ
cond.mean.eta	Estimate of the corresponding random effect
cond.var.phi	Estimate of the variance of the individual ϕ
cond.shrinkage	Shrinkage for the conditional estimates
phi.samp	Samples from the individual conditional distribution of the ϕ
phi.samp.var	Variance of the samples from the individual conditional distribution of the ϕ
ypred	Population predictions, computed for the mean population parameters $ypred_{ij} = f\left(x_{ij}; h\left(\mathbb{E}_{\hat{\theta}}(\phi_i)\right)\right)$
ppred	Population mean predictions, obtained as the expectation of the predictions $ppred_{ij} = \mathbb{E}_{\hat{\theta}}(f(x_{ij}; \psi_i)))$
ipred	Individual predictions, computed using the MAP estimates of the individual parameters
icpred	Individual predictions, computed using the conditional estimates of the individual parameters
wres	Weighted population residuals, computed using ppred (see section 2.6)
pd	Prediction discrepancies

– To be continued

Table 2.2 – cont.

Element	Meaning
npde	Normalised prediction distribution errors
iwres	Individual weighted residuals, using the MAP estimates of the individual parameters (using the same computations as ipred)
icwres	Individual weighted residuals using the conditional estimates of the individual parameters (using the same computations as icpred)

Table 2.2: Elements contained in the results object.

A full list of all the elements in a results object can be obtained by the command:

```
getSlots("SaemixRes")
```

a) Estimation of the parameters:

The SAEM algorithm computes the maximum likelihood estimate $\widehat{\theta}$ and estimates its covariance matrix $I(\widehat{\theta})^{-1}/N$ using the Fisher Information Matrix, as defined in Section 2.2.

Recall that d is the number of individual parameters, then for $j = 1, 2 \dots d$, we can

1. estimate the vector of fixed effects μ (intercept and coefficients of the covariates) by $(\hat{\mu})$,
2. estimate the standard errors of μ,
3. test if some components of μ are null by computing the significance level of the Wald test.

Let $\Omega = (\omega_{jl}, 1 \leq j, l \leq d)$. Then, for any $j, l = 1, 2 \dots d$, we can

1. estimate ω_{jl} by $\widehat{\omega}_{jl}$, for all $1 \leq j, l \leq d$,
2. estimate the standard error of $\widehat{\omega}_{jl}$, for all $1 \leq j, l \leq d$,

b) Estimation of the conditional distributions:

The MCMC algorithm provides an estimation of the conditional means, conditional modes and conditional standard deviations of the individual parameters and of the random effects.

The function can be called with an argument `nsamp` which runs several sampling chains in parallel, providing several independent samples from the individual conditional distribution for each subject. The number of iterations necessary to obtain convergence (that is, for the successive empirical conditional mean and sd to remain within the requested precision for all chains) is reported, and if the option `displayProgress` is `TRUE`, plots are produced during the estimation process showing the evolution of the different sampling chains.

- the conditional mode can be found in SAEMIX in the results component of the object, as `map.psi` (there is also a `map.phi` component for the corresponding ϕ and a `map.eta` for the random effects)
- the conditional expectation can be found in `cond.mean.psi` and the variance in `cond.var.psi` (the corresponding ϕ and η are also available)

c) Estimation of the likelihood:

The SAEMIX algorithm can provide three different approximations to the likelihoods, through importance sampling, linearisation or gaussian quadrature.

d) Hypothesis testing and model selection:

We can test the covariate model, the covariance model and the residual error model.

The AIC and BIC criteria are defined by

$$AIC = -2\log \ell_M(y;\widehat{\theta}) + 2P \quad (2.27)$$
$$BIC = -2\log \ell_M(y;\widehat{\theta}) + \log(N)P \quad (2.28)$$

where P is the total number of parameters to be estimated and N is the number of subjects. Note that the BIC defined using this formula is in fact the corrected BIC (BICc) proposed by Raftery to better account for the information in mixed-effect models [25]; it differs from the traditional BIC which uses a factor $\log(N_{tot})$ instead of $\log(N)$. The same formula is also used in MONOLIX.

When comparing two nested models $\mathcal{M}_0$ and $\mathcal{M}_1$ with dimensions P_0 and P_1 (with $P_1 > P_0$), the Likelihood Ratio Test uses the test statistic

$$LRT = 2(\log \ell_{M,1}(y;\widehat{\theta}_1) - \log \ell_{M,0}(y;\widehat{\theta}_0))$$

According to the hypotheses to test, the limiting distribution of LRT under the null hypothesis is either a χ^2 distribution, or a mixture of a χ^2 distribution and a $\delta - Dirac$ distribution. For example:

- to test whether some fixed effects are null, assuming the same covariance structure of the random effects, one should use

$$LRT \underset{N\to\infty}{\longrightarrow} \chi^2(P_1 - P_0)$$

- to test whether some correlations of the covariance matrix Ω are null, assuming the same covariate model, one should use

$$LRT \underset{N\to\infty}{\longrightarrow} \chi^2(P_1 - P_0)$$

- to test whether the variance of one of the random effects is zero, assuming the same covariate model, one should use

$$LRT \underset{N\to\infty}{\longrightarrow} \frac{1}{2}\chi^2(1) + \frac{1}{2}\delta_0$$

e) Estimation of the weighted residuals:

The Population Weighted Residuals ($PWRES_{ij}$), the Individual Weighted Residuals ($IWRES_{ij}$) and the Normalised Prediction Distribution Errors (npde_{ij}) are computed as described Section 2.6.

2.7.3 Plots

The generic function plot.saemix can be used to obtain a number of plots used to assess and diagnose the model. This function is called using the following arguments:

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

where saemix.fit is the object returned after a successful call to saemix, and "plot.type" is the type of plot chosen. The following plot types are available:

- "data": spaghetti plot of the data
- "convergence": a plot of the convergence graphs; this is the default type when type is not given
- "likelihood": estimate of the likelihood through importance sampling versus the number of MCMC samples
- "individual.fit": plot of the individual fits overlayed on the data, for each subject in the dataset
- "population.fit": plot of the fits obtained with the population parameters and the individual covariates and design, overlayed on the data, for each subject in the dataset
- "both.fit": plot of the individual and population fits, overlayed on the data
- "observations.vs.predictions": observations versus predictions(left: population predictions, right: individual predictions)
- "random.effects": boxplot of the random effects. With the option "m", a horizontal line is added representing the estimate of the population parameter
- "parameters.versus.covariates": plot of a parameter versus all covariates in the model (uses the individual estimates); for continuous covariates, a scatterplot is produced, while for categorical covariates a boxplot is shown. With the option "m", a horizontal line is added representing the estimate of the population parameter. With the options "l" or "s", a curve representing a linear regression ("l") or a spline regression ("s") is added. Several options can be combined (see below)
- "randeff.versus.covariates": plot of a random effect versus all covariates in the model (uses the individual estimates)
- "correlations": matrix of scatterplot showing the correlations between pairs of random effects (uses the individual estimates)

- "marginal.distribution": distribution of the random effects

- "residuals.distribution": distribution of the standardised residuals, computed using the population predictions (weighted residuals), the individual predictions (individual weighted residuals) and optionally if available the npde. Both histograms and QQ-plots of the residuals are given

- "residuals.scatter": scatterplot of standardised residuals versus the predictor (X) and versus the predictions. The residuals are computed using the population predictions (weighted residuals), the individual predictions (individual weighted residuals) and optionally if available the npde. The corresponding predictions are the individual predictions for individual residuals, and population predictions for npde and population residuals

- "vpc": Visual Predictive Check; prediction intervals can be added to the plots. To produce prediction intervals, different methods are available for binning (grouping points), which can be selected through the vpc.method argument:

  equal: the quantile of the data are used to define the breaks, yielding a similar number of points in each interval;

  width: bins of equal width (if the option xlog is set to TRUE, the bins will be of equal width on the logarithmic scale);

  user: user-defined breaks (set as the vector in vpc.breaks argument; it is possible to give only the inner breaks or to include the boundaries (min/max));

  In the first three methods, there will be at most vpc.bin bins, and the boundaries of each interval, as well as the value used to plot the corresponding point, will be shown.

- "npde": plots of the npde (distribution, histogram, and scatterplots versus the regression variable and versus predictions), as displayed in the npde library [2]. Tests comparing the empirical distribution of the npde to the theoretical $\mathcal{N}(0,1)$ distribution by a combined test are also displayed.

Several plots can be produced by setting plot.type to be a vector. Partial matching will be used (so that plot.type="individual" will produce individual fits, but plot.type="residuals" will produce an error message because it could correspond to two different types of plots). After a successful fit, if the option save.graphs is TRUE, the following plots are produced by default and saved to a file named diagnostic_graphs.ps in the directory containing fit results: spaghetti plot of the data, convergence plots, likelihood by importance sampling, plots of predictions versus observations for population and individual estimates, boxplots of the random effects, correlation between the random effects. Individual fits are also saved, in a separate file called individual_fits.ps. Some of these plots may be missing if the corresponding estimates have not been requested (eg if the likelihood has not been computed by importance sampling, the plot won't be available).

Each plot can also be obtained individually using a specific function, which allows total flexibility over the layout, including options to change plotting symbols, colors, or which subjects are to be used. Table 2.3 gives the names of the individual functions corresponding to the plots listed above.

Plot function name	Brief description
saemix.plot.data()	Spaghetti plot of the data
saemix.plot.convergence()	Convergence plots for all estimated parameters
saemix.plot.llis()	Plot of the log-likelihood estimated by importance sampling
saemix.plot.obsvspred()	Plot of the predictions versus the observations
saemix.plot.fits()	Individual fit
saemix.plot.distpsi()	Estimated distribution of the random effects
saemix.plot.randeff()	Boxplot of a random effect
saemix.plot.parcov()	Plot of parameters versus covariates
saemix.plot.randeffcov()	Plot of random effects versus covariates
saemix.plot.scatterresiduals()	Scatterplots of residuals versus predictor and predictions
saemix.plot.distribresiduals()	Plot of the distribution of the residuals
saemix.plot.vpc()	Visual Predictive Check
saemix.plot.npde()	Plots of the npde

Table 2.3: Names of the individual functions used to obtain each type of plot. Please refer to the inline help for the arguments to provide to each function.

A help page describing these plots is available in the inline help:

```
?saemix.plot.data
```

A common argument to all the functions is a list of options. This list can be set using the function saemix.plot.setoptions(), and it is automatically set during the fit by saemix() and stored in the Slot prefs of the object. The options can then be modified through this list, for instance changing the new default color to red for all plots is done by setting the attribute col in the list:

```
saemix.fit["prefs"]$col<-"red"
```

Options can also be set on the fly for a given plot, by simply adding it to the call to plot() as an argument (see examples in section 3.1):

```
plot(saemix.fit,plot.type="data",col="red",main="Raw data")
```

The list of options that can be changed are given in table 2.4, along with their default value. Not all options apply to all graphs.

Parameter	Description	Default value
General graphical options		
ask	Whether users should be prompted before each new plot (if TRUE)	FALSE
new	Whether a new plot should be produced	TRUE
interactive	Whether users should be prompted before predictions or simulations are performed (if TRUE)	FALSE
mfrow	Page layout (NA: layout set by the plot function or before)	NA
main	Title	empty
xlab	Label for the X-axis	empty
ylab	Label for the Y-axis	empty
type	Type of the plot (as in the R plot function)	b (lines and symbols)
col	Main symbol color	black
xlog	Scale for the X-axis (TRUE: logarithmic scale)	FALSE
ylog	Scale for the Y-axis (TRUE: logarithmic scale)	FALSE
cex	A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default	1
cex.axis	Magnification to be used for axis annotation relative to the current setting of 'cex'	1
cex.lab	Magnification to be used for x and y labels relative to the current setting of 'cex'	1
cex.main	Magnification to be used for main titles relative to the current setting of 'cex'	1
pch	Symbol type	20 (dot)
lty	Line type	1 (straight line)
lwd	Line width	1
xlim	Range for the X-axis (NA: ranges set by the plot function)	NA
ylim	Range for the Y-axis (NA: ranges set by the plot function)	NA
ablinecol	Color of the horizontal/vertical lines added to the plots	"DarkRed"
ablinelty	Type of the lines added to the plots	2 (dashed)
ablinelwd	Width of the lines added to the plots	2
Options controlling the type of plots		
ilist	List of subjects to include in the individual plots	all

– To be continued

Table 2.4 – cont.

Parameter	Description	Default value
level	Level of grouping to use (0=population, 1=individual)	0:1
smooth	Whether a smooth should be added to certain plots	FALSE
line.smooth	Type of smoothing (l=line, s=spline)	s
indiv.par	Type of individual estimates (map= conditional mode, eap=conditional mean)	map
which.par	Which parameters to use for the plot	all
which.cov	Which covariates to use for the plot	all
which.pres	Which type of residuals to plot at the population level (when level includes 0)	c(”wres”,”npde”)
which.resplot	Type of residual plot (”res.vs.x”: scatterplot versus X, ”res.vs.pred”: scatterplot versus predictions, ”dist.hist”: histogram, ”dist.qqplot”: QQ-plot)	c(”res.vs.x”,”res.vs.pred”, ”dist.qqplot”,”dist.hist”)
Specific graphical options		
obs.col	Symbol color to use for observations	black
ipred.col	Symbol color to use for individual predictions	black
ppred.col	Symbol color to use for population predictions	black
obs.lty	Line type to use for observations	1
ipred.lty	Line type to use for individual predictions	2
ppred.lty	Line type to use for population predictions	3
obs.lwd	Line width to use for observations	1
ipred.lwd	Line width to use for individual predictions	1
ppred.lwd	Line width to use for population predictions	1
obs.pch	Symbol type to use for observations	20
ipred.pch	Symbol type to use for individual predictions	20
ppred.pch	Symbol type to use for population predictions	20
Options for marginal distribution		
indiv.histo	When TRUE, an histogram of the estimates of the individual parameters will be added to the plots of the distribution of the parameters	FALSE
cov.value	The value for each covariate to be used to condition on for the marginal distribution (NA: median will be used)	NA
range	Range (expressed in number of SD) over which to plot the marginal distribution	3

– To be continued

Table 2.4 – cont.

Parameter	Description	Default value
Graphical options for VPC and residual plots		
vpc.method	Method used to bin points (one of "equal", "width", "user" or "optimal"); at least the first two letters of the method need to be specified (the "optimal" method is not implemented yet)	"equal"
vpc.bin	number of binning intervals	10
vpc.interval	size of interval	0.95
vpc.breaks	vector of breaks used with user-defined breaks (vpc.method="user")	NULL
vpc.lambda	value of lambda used to select the optimal number of bins through a penalised criterion	0.3
vpc.pi	whether prediction intervals should be plotted for the median and the limits of the VPC interval	TRUE
vpc.obs	whether observations should be overlayed on the plot	TRUE
fillcol	Color used to fill histograms (individual parameter estimates) or to plot intervals in standard VPC-type plots (VPC, pd, npde)	"lightblue1"
col.fillmed	Color used to fill prediction intervals around the median (for VPC, pd, npde)	"pink"
col.fillpi	Color used to fill prediction intervals around the limits of intervals (for VPC, pd, npde)	"slategray1"
col.lmed	Color used to plot the median of simulated values (for VPC, pd, npde)	"indianred4"
col.lpi	Color used to plot the simulated limit of prediction intervals (for VPC, pd, npde)	"slategray4"
col.pobs	Color used to plot the symbols for observations (for VPC, pd, npde)	"steelblue4"
col.lobs	Color used to plot the line corresponding to given percentiles of observations (for VPC, pd, npde)	"steelblue4"
lty.lmed	Line type used to plot the median of simulated values (for VPC, pd, npde)	2
lty.lpi	Line type used to plot the simulated limit of prediction intervals (for VPC, pd, npde)	2
lty.lobs	Line type used to plot the line corresponding to given percentiles of observations (for VPC, pd, npde)	1

– To be continued

Table 2.4 – cont.

Parameter	Description	Default value
lwd.lmed	Line width used to plot the median of simulated values (for VPC, pd, npde)	2
lwd.lpi	Line width used to plot the simulated limit of prediction intervals (for VPC, pd, npde)	1
lwd.lobs	Line width used to plot the line corresponding to given percentiles of observations (for VPC, pd, npde)	2
Specific graphical options		
pcol	Main symbol color	black
lcol	Main line color	black

Table 2.4: Default graphical parameters. Any option not defined by the user is automatically set to its default value.

Chapter 3

Examples

3.1 Theophylline pharmacokinetics

3.1.1 One-compartment model

Boeckmann, Sheiner and Beal (1994) report data from a study by Dr. Robert Upton of the kinetics of the anti-asthmatic drug theophylline [31]. 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 the present package, we removed the data at time 0 to avoid some unexplained non-zero values in a supposedly single-dose study. In the original dataset shipped with the NONMEM, doses are given as doses per kilo body weight. Here, we therefore also transformed the doses to doses in mg instead of mg/kg.

These data are analyzed in Davidian and Giltinan [4] and Pinheiro and Bates [21] using a two-compartment open pharmacokinetic model. These data are also available in R in the library datasets under the name Theoph in a slightly modified format and including the data at time 0.

Subject i receives an initial dose D_i at time 0 and serum concentrations y_{ij} are measured at time t_{ij}. Serum concentration is modeled by a first-order one compartment model, according to the following equation:

$$y_{ij} = \frac{D_i k_{ai} k_{ei}}{CL_i(k_{ai} - k_{ei})} \left(e^{-k_{ai}t_{ij}} - e^{-k_{ei}t_{ij}} \right) + \epsilon_{ij} \tag{3.1}$$

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 $k_{ei} = \frac{CL_i}{V_i}$. For subject i:

- the vector of regression (or design) variables is $x_{ij} = (D_i, t_{ij})$
- the vector of individual parameters is $\theta_i = (\ln(k_{ai}), \ln(CL_i), \ln(V_i))$
  - k_{ai}, CL_i and V_i are assumed to be independant log-normal random variables
  - we assumed a relationship between the clearance and the subject's body weight BW_i

$$\begin{aligned} \ln(k_{ai}) &= \mu_1 + \eta_1 \\ \ln(V_i) &= \mu_2 + \eta_2 \\ \ln(CL_i) &= \mu_3 + \beta \mathrm{BW}_i + \eta_3 \end{aligned} \tag{3.2}$$

- we can use a simple homoscedastic error model where $\mathrm{Var}\,(\epsilon_{ij}) = a^2$

The data is shown in figure 3.1.

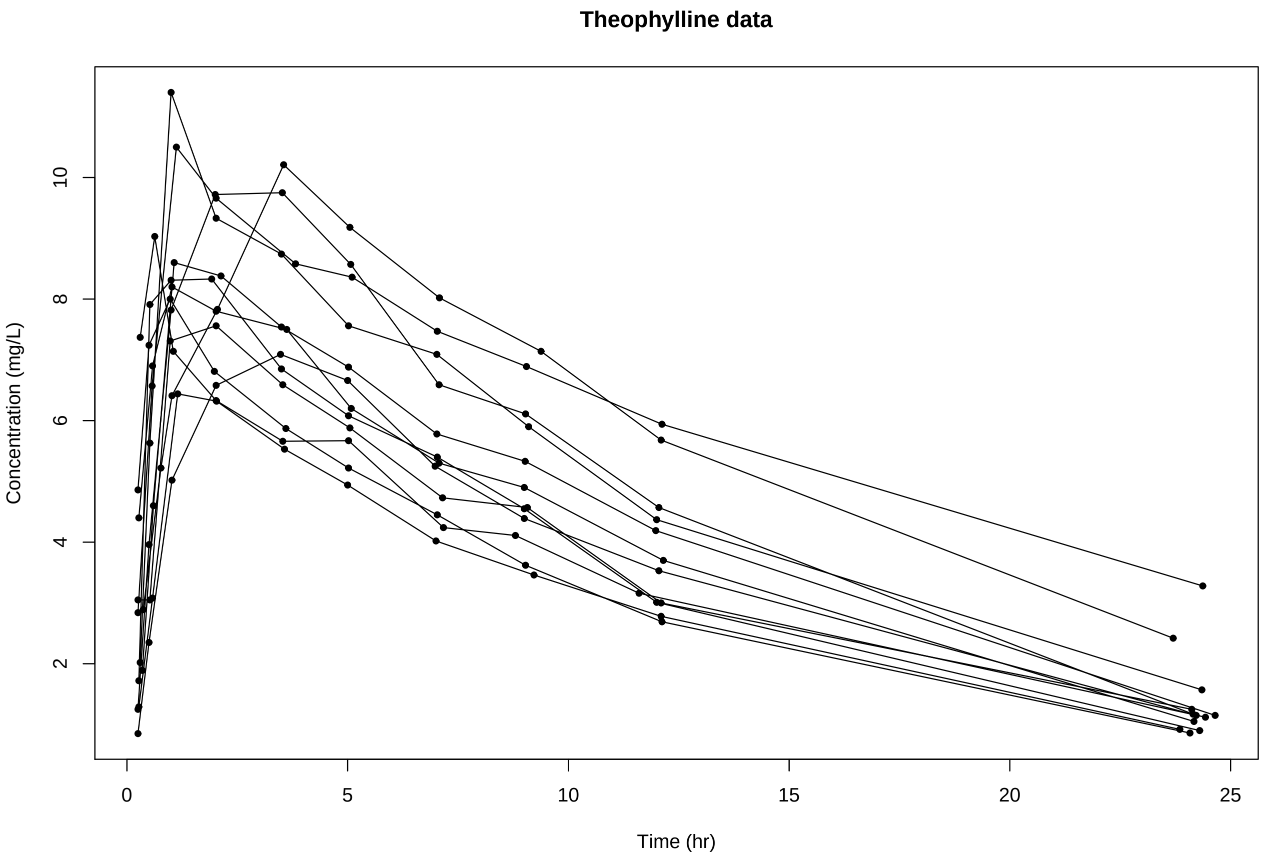

Figure 3.1: Theophylline concentrations versus time for the 12 subjects included in the study.

The following code was used in R to read the data and model:

```
library(saemix)

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

model1cpt<-function(psi,id,xidep) {
  dose<-xidep[,1]
  tim<-xidep[,2]
  ka<-psi[id,1]
  V<-psi[id,2]
  CL<-psi[id,3]
  k<-CL/V
  ypred<-dose*ka/(V*(ka-k))*(exp(-k*tim)-exp(-ka*tim))
  return(ypred)
}
saemix.model<-saemixModel(model=model1cpt,
description="One-compartment model with first-order absorption",
psi0=matrix(c(1.,20,0.5,0.1,0,-0.01),ncol=3, byrow=TRUE,dimnames=list(NULL,
c("ka","V","CL"))),transform.par=c(1,1,1),
covariate.model=matrix(c(0,0,1,0,0,0),ncol=3,byrow=TRUE),fixed.estim=c(1,1,1),
covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE), error.model="constant")
```

In this example, we specify the vector of starting values through `psi0`, which is defined as the following matrix:

```
      ka  V    CL
[1,] 1.0 20  0.50
[2,] 0.1  0 -0.01
```

The first line is renamed as Pop.CondInit when the model object is created (see output from the commands given in the snippet of code above), and contains the initial estimates of the population parameters k_a, V and CL. The second line, renamed Cov.CondInit in this example, contains the initial values for the parameter-covariate relationships in the model. In this example, we have assumed an effect of the covariate Weight on the clearance CL, and the initial value of the corresponding fixed effect is -0.01. In this model there is no relationship between either of the two covariates in the model and k_a, so that the 0.1 value given in `psi0` will not be used. If we also had relationships between the covariate Sex and the model parameters, the same starting values would be used (using the vector recycling principle R), however we could add other lines to `psi0` to specify different starting values. For example, assuming we want to estimate an effect of weight on V and CL, as well as a gender effect on CL, we could replace the `covariate.model` argument with:

```
covariate.model=matrix(c(0,1,1,0,0,1),ncol=3,byrow=TRUE)
```

and give different starting values for each parameter-relationship in `psi0`, for example 0.1 for both weight effects and -0.1 for the gender effect:

```
psi0=matrix(c(1.,20,0.5,0,0.1,0.1,0,0,-0.1),ncol=3, byrow=TRUE,dimnames=list(NULL,
c("ka","V","CL")))
```

Note that the model requires two predictors, dose and time. The user is responsible for writing the model function and checking the consistency between the model function and the data. Here, the first predictor (first column) is dose and the second predictor is time so that we need both items in the dataset, and we need to give the names of the two predictors in the proper order (the order corresponding to the way the model function is written here) when creating the data object. This is a single-dose administration and therefore the dose column contains the same dose repeated for each time-point. However, for graphs we want the observations to be plotted versus time and not versus dose; by default, the program will use the first predictor as the X axis, but we override this behaviour here by setting the option name.X="Time" in the creation of the data object, so that the graphs will use time on the X-axis.

Then we fit the model using the saemix() function:

```
saemix.fit<-saemix(saemix.model,saemix.data,list(seed=632545,nb.chains=5,
nbiter.saemix = c(300, 150)))
```

We use 5 chains here to stabilise the estimation because there are only 12 subjects in the dataset (by default, the algorithm will increase the number of chains if there are less than 50 subjects in the dataset, and set it to a higher value as describe in section 2.7), and we increase the number of steps in the second stage to 150 (default: 100) to show how to set this option. Increasing the number of iterations in the second stage helps to obtain a more stable conditional distribution for the individual parameters.

This produces the following output:

```
 ............
Nonlinear mixed-effects model fit by the SAEM algorithm
-----------------------------------
```

```
----          Data             ----
-----------------------------------
Object of class SaemixData
    longitudinal data for use with the SAEM algorithm
Dataset theo.saemix
    Structured data: Concentration ~ Dose + Time | Id
    X variable for graphs: Time (hr)
    covariates: Weight (kg), Sex (-)
First 10 lines of data:
   Id    Dose  Time Concentration Weight Sex
1   1 319.992  0.25          2.84   79.6   1
2   1 319.992  0.57          6.57   79.6   1
3   1 319.992  1.12         10.50   79.6   1
4   1 319.992  2.02          9.66   79.6   1
5   1 319.992  3.82          8.58   79.6   1
6   1 319.992  5.10          8.36   79.6   1
7   1 319.992  7.03          7.47   79.6   1
8   1 319.992  9.05          6.89   79.6   1
9   1 319.992 12.12          5.94   79.6   1
10  1 319.992 24.37          3.28   79.6   1
-----------------------------------
----          Model            ----
-----------------------------------
Nonlinear mixed-effects model
  Model function:  One-compartment model with first-order absorption
function(psi,id,xidep) {
  dose<-xidep[,1]
  tim<-xidep[,2]
  ka<-psi[id,1]
  V<-psi[id,2]
  CL<-psi[id,3]
  k<-CL/V
  ypred<-dose*ka/(V*(ka-k))*(exp(-k*tim)-exp(-ka*tim))
  return(ypred)
}
  Nb of parameters: 3
      parameter names:  ka V CL
      distribution:
     Parameter Distribution
[1,] ka        log-normal
[2,] V         log-normal
[3,] CL        log-normal
  Variance-covariance matrix:
```

```
   ka V CL
ka  1 0  0
V   0 1  0
CL  0 0  1
  Error model: constant , initial values: a=1
  Covariate model:
       ka V CL
Weight  0 0  1
    Initial values
       ka  V    CL
PopCI 1.0 20  0.50
CovCI 0.1  0 -0.01
-----------------------------------
----    Key algorithm options  ----
-----------------------------------
    Algorithms: MAP, FIM, LL by IS
    Number of iterations:  K1=300, K2=150
    Number of chains:  5
    Seed:  632545
    Number of MCMC iterations for IS:  5000
    Simulations:
        nb of simulated datasets used for npde:  1000
        nb of simulated datasets used for VPC:  100
    Input/output
        save the results to file pop_parameters.txt in directory:  newdir
        save graphs
-----------------------------------
----          Results          ----
-----------------------------------
----------------------------------------------------
-----------------  Fixed effects  ------------------
----------------------------------------------------
     Parameter       Estimate SE     CV(%) p-value
[1,] ka               1.567   0.2998  19.1 -
[2,] V               31.475   1.3838   4.4 -
[3,] CL               1.581   1.0155  64.2 -
[4,] beta_Weight(CL)  0.008   0.0092 113.5 0.19
[5,] a                0.743   0.0569   7.7 -
----------------------------------------------------
-----------  Variance of random effects  -----------
----------------------------------------------------
   Parameter Estimate SE    CV(%)
ka omega.ka  0.388    0.175 45
```

```
V  omega.V   0.015    0.009 59
CL omega.CL  0.070    0.034 49
----------------------------------------------------
------  Correlation matrix of random effects  ------
----------------------------------------------------
         omega.ka omega.V omega.CL
omega.ka 1        0       0
omega.V  0        1       0
omega.CL 0        0       1
----------------------------------------------------
---------------  Statistical criteria  -------------
----------------------------------------------------
Likelihood computed by linearisation
      -2LL= 343.4919
      AIC = 359.4919
      BIC = 363.3712

Likelihood computed by importance sampling
      -2LL= 344.8896
      AIC = 360.8896
      BIC = 364.7689
----------------------------------------------------
```

By default, the results are saved in a file called pop_parameters.txt in the newdir directory, and graphs are produced.

Table 3.1 reports the parameters obtained on a Linux Ubuntu distribution running R version 2.11.1 for this example. In this example, the fixed effect representing the influence of weight on CL is not significant (p=0.19, NS according to a Wald test).

Parameter	Population estimate (SE%)	IIV Variance (SE%)
k_a (hr^{-1})	1.57 (19%)	0.39 (45%)
CL (L.hr^{-1})	1.58 (64%)	0.07 (49%)
$\beta_{BW,CL}$ (-)	0.008 (110%)	-
V (L)	31.5 (4%)	0.02 (59%)
a (mg.L^{-1})	0.74 (6%)	-

Table 3.1: Pharmacokinetic parameters estimated by SAEMIX for the theophylline data.

A series of diagnostic plots can be produced simply by applying the function plot() to the object returned by saemix():

```
plot(saemix.fit)
```

By using the plot.type="" argument, specific graphs can be produced (see section 2.7.3). For example, the convergence plot shown in figure 3.2 can be produced by:

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

In this figure we can see all the parameters converging quickly to their estimated value.

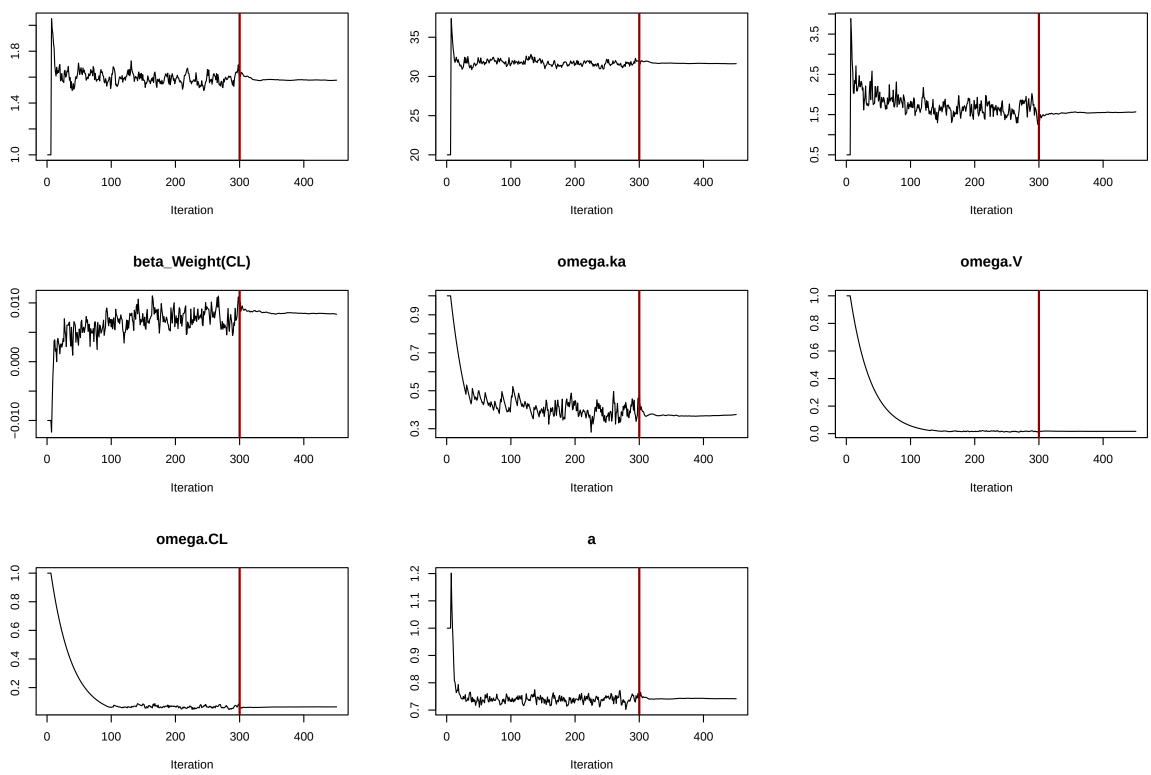

Figure 3.2: Convergence plots for the estimated pharmacokinetic parameters and the variabilities.

Figure 3.3 shows the evolution of the log-likelihood during the importance sampling step. Figure 3.4 shows the predicted values compared to the observed concentrations, for the population predictions (left) and the individual predictions (right). Figure 3.5 shows the individual data for the 12 subjects, with the individual predictions overlayed (smoothed predictions were obtained). Both plots indicate good model adequacy.

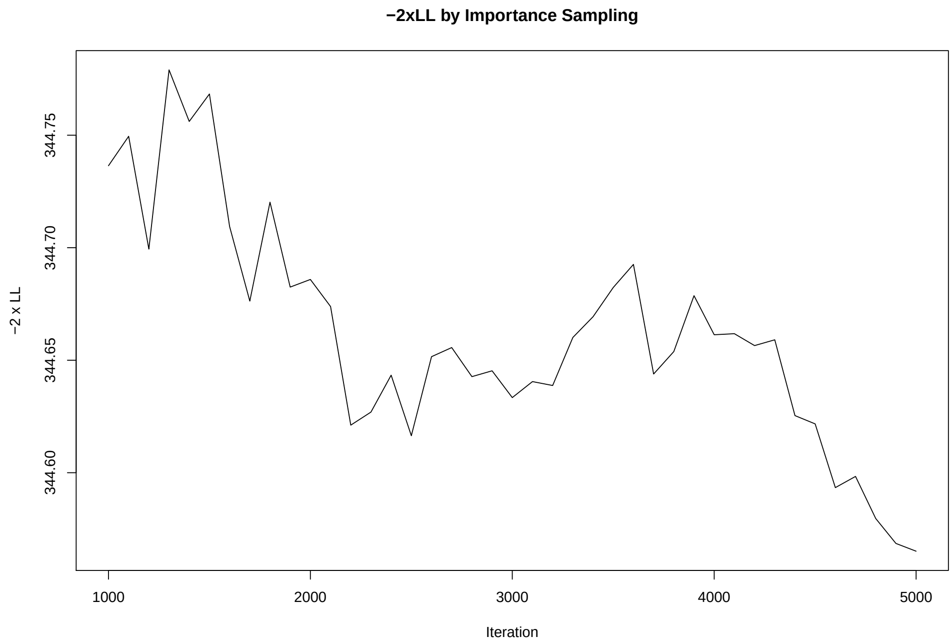

Figure 3.3: Estimating the log-likelihood by Importance Sampling.

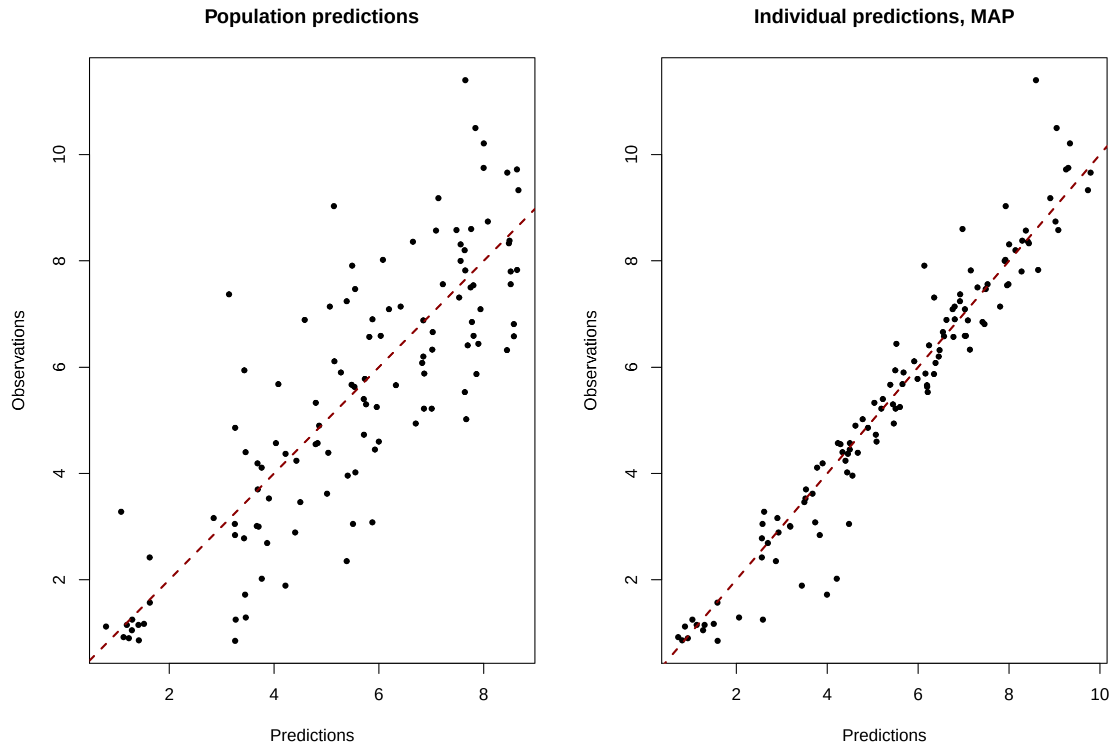

Figure 3.4: Observations versus predictions (left: population predictions, right: individual predictions).

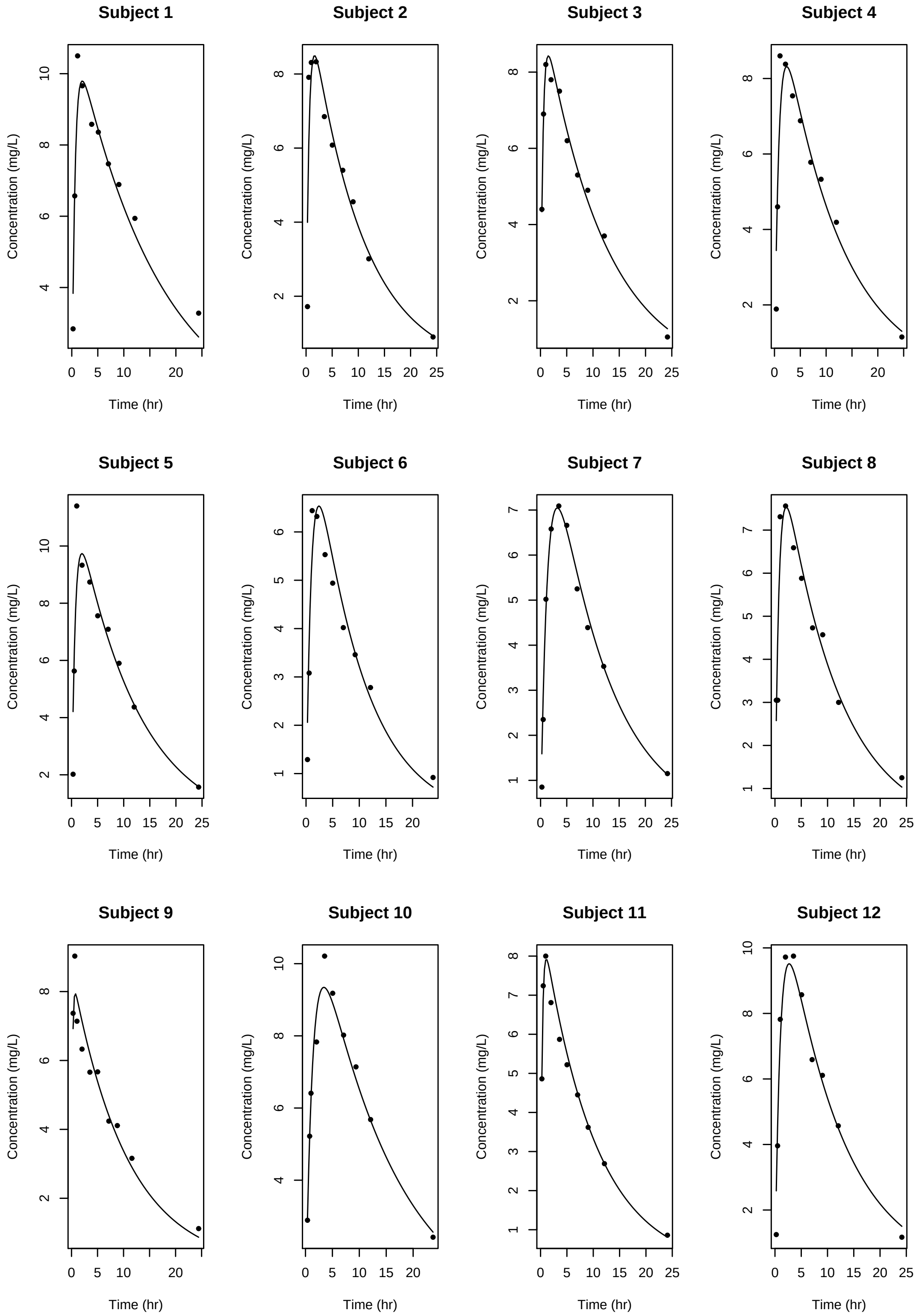

Figure 3.5: Individual plots for the 12 subjects in the study. Dots represent observations and the line shows the smoothed profile predicted using the individual estimated parameters.

The following example shows how to use the functions defined in section 2.7.3 to plot 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. The resulting plot is shown in figure 3.6

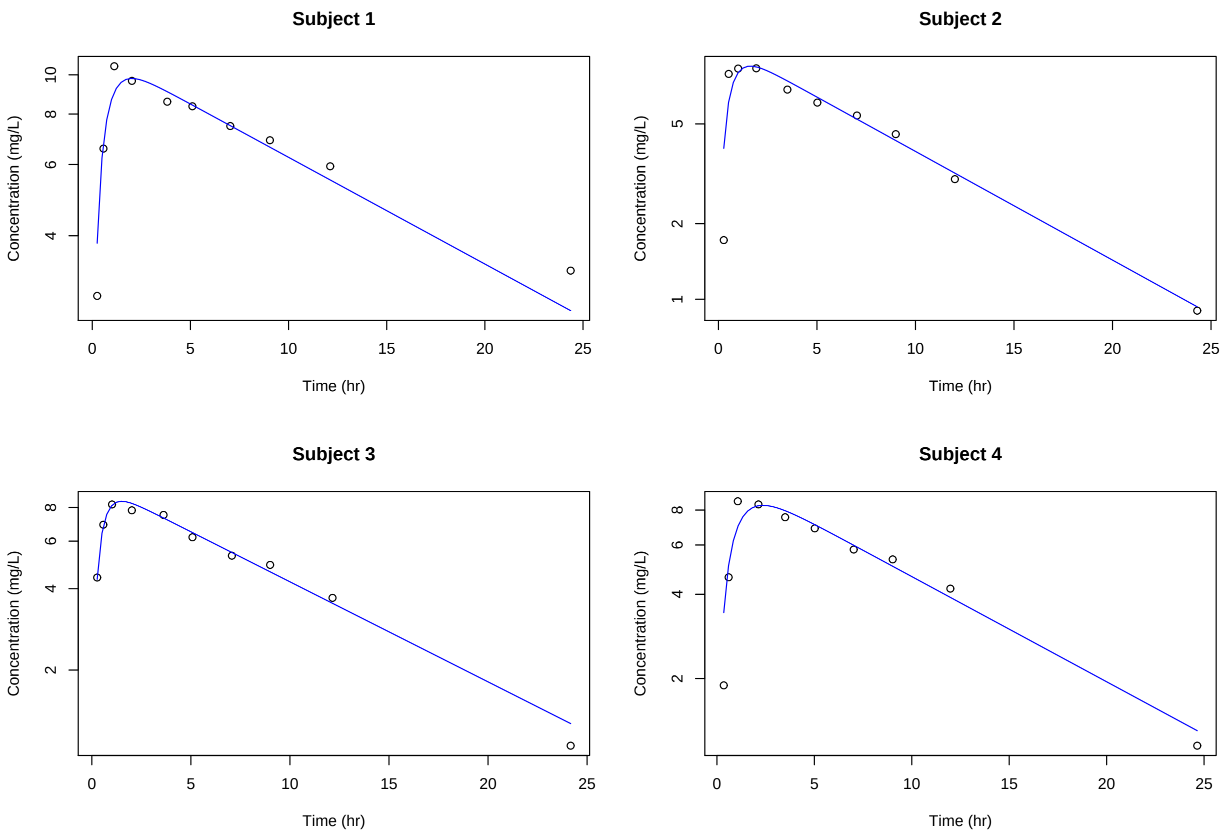

Figure 3.6: Individual plots for the first 4 subjects in the study, with different options.

To obtain these plots, we can use the generic function plot(), by setting the plot.type argument to "individual.fit", to produce these plots:

```
# Plotting individual fits with selected options
par(mfrow=c(2,2))
plot(saemix.fit,plot.type="individual.fit",new=FALSE,ilist=1:4,smooth=TRUE,ylog=T,
pch=1, col="Blue",xlab="Time in hr",ylab="Theophylline concentrations (mg/L)")
```

We can also use directly the saemix.plot.fits() function with the same graphical options, which gives the exact same graph:

```
# Plotting individual fits with selected options
par(mfrow=c(2,2))
saemix.plot.fits(saemix.fit,new=FALSE,ilist=1:4,smooth=TRUE,ylog=T,pch=1,
col="Blue",xlab="Time in hr",ylab="Theophylline concentrations (mg/L)")
```

Other diagnostic plots include Visual Predictive Checks, shown in figure 3.7, and residual plots.

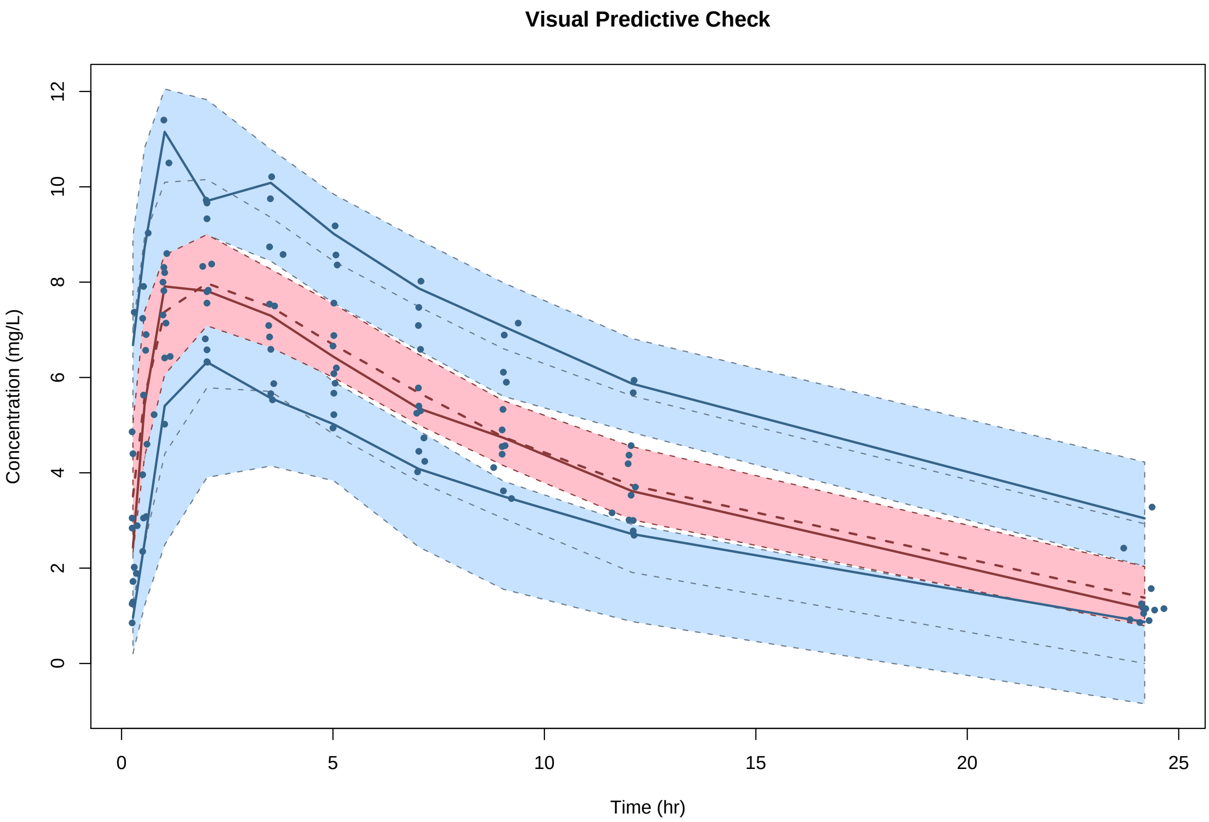

Figure 3.7: VPC for the theophylline data.

The following code can be used to first simulate from the model in order to compute simulation-based metrics (residuals and VPC), and then produce VPC and scatterplots of residuals versus time and predictions (figure 3.8).

```
# Scatterplots of residuals
plot(saemix.fit, plot.type="residuals.scatter")

# VPC
```

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

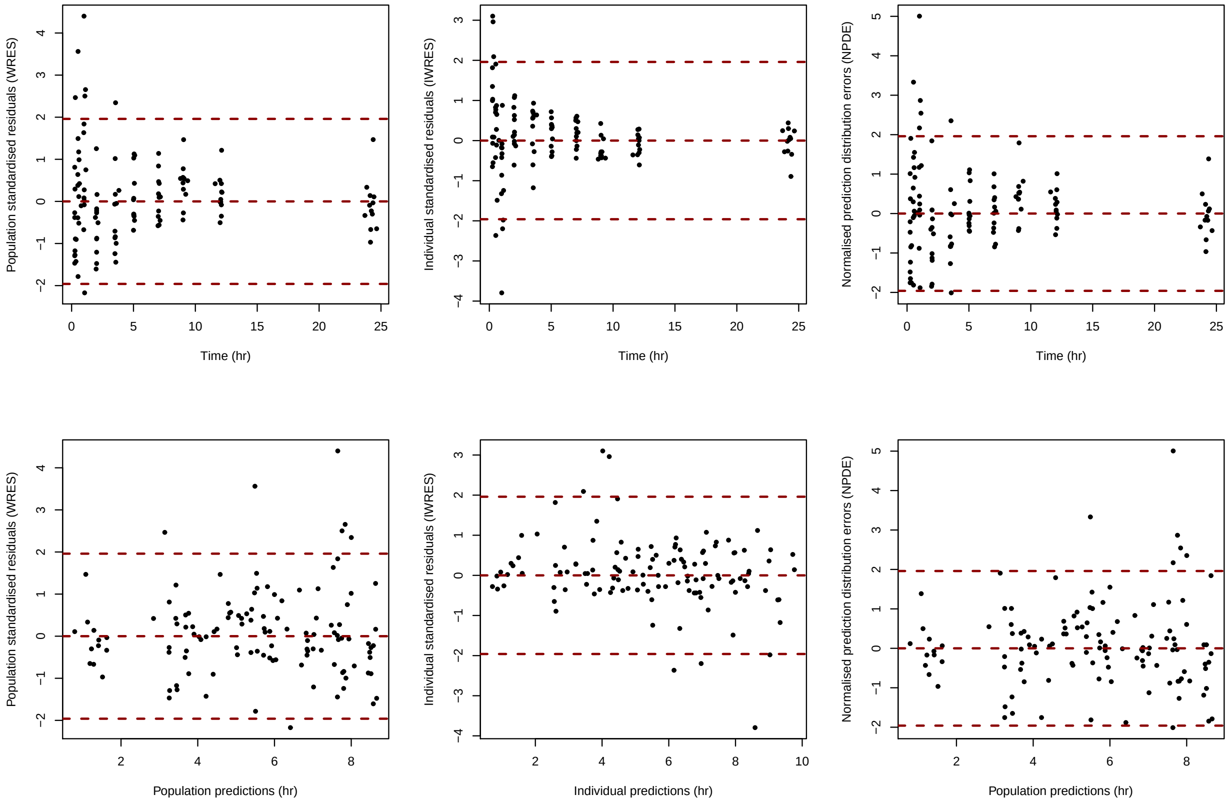

Figure 3.8: Scatterplots of the residuals (left: weighted population residuals; middle: individual weighted residuals; right: npde) versus time (top) and predictions (bottom).

Finally, note that the SAEM algorithm is relatively robust to the initial choice of parameter estimates, but different initial choices may lead to different population estimates. Here, if we had set all the initial parameters to 1 as in the following code, the model converges to very different values and a flip-flop occurrs (k_a becomes smaller than the elimination rate constant $k = CL/V$). The resulting fit however has a lower likelihood and the VPC graphs indicate poor estimates of the variability (not shown), which can give an indication of problems with the model.

```
saemix.model<-saemixModel(model=model1cpt,
description="One-compartment model with first-order absorption",
psi0=matrix(c(1.,1.,1.,0.1,0,-0.01),ncol=3, byrow=TRUE,dimnames=list(NULL,
c("ka","V","CL"))),transform.par=c(1,1,1),
covariate.model=matrix(c(0,0,1,0,0,0),ncol=3,byrow=TRUE), fixed.estim=c(1,1,1),
covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
```

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

Thus, it is always good policy during data analysis to check the stability of the final model estimates by changing the initial estimates and running the algorithm again, and to compare the magnitude of the parameter estimates with a reference, such as prior information or litterature values.

3.1.2 One-compartment model at steady-state

In the theophylline example, we described the pharmacokinetics using the single-dose, first-order absorption and elimination model. The following code shows how to fit the same data with the same model at steady-state, assuming a 24 hours dosing interval:

```
data(theo.saemix)
# Include a column for the inter-dose interval (tau)
theo.saemix2<-cbind(theo.saemix,tau=24)
saemix.data2<-saemixData(name.data=theo.saemix2,header=TRUE,sep=" ",na=NA,
  name.group=c("Id"),name.predictors=c("Dose","Time","tau"),
  name.response=c("Concentration"),name.covariates=c("Weight","Sex"),
  units=list(x="hr",y="mg/L",covariates=c("kg","-")), name.X="Time")
# Define the model for steady-state
modelSS<-function(psi,id,xidep) {
   dose<-xidep[,1]
   tim<-xidep[,2]
   tau<-xidep[,3]
   ka<-psi[id,1]
   V<-psi[id,2]
   CL<-psi[id,3]
   k<-CL/V
   ypred<-dose*ka/(V*(ka-k))*(exp(-k*tim)/(1-exp(-k*tau))-
exp(-ka*tim)/(1-exp(-ka*tau)))
   return(ypred)
}
saemix.model2<-saemixModel(model=modelSS,
  description="One-compartment model with first-order absorption, Steady-state",
  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))

# Run SAEMIX again
saemix.options<-list(seed=632545)
saemix.fit2<-saemix(saemix.model2,saemix.data2,saemix.options)
```

3.2 Simulated pharmacodynamic model

A symposium dedicated to Comparison of Algorithms Using Simulated Data Sets and Blind Analysis, took place in Lyon, France, September 2004, organised by P. Girard and F. Mentré. During this symposium, a blind comparison of several PK/PD modelling software was performed, using simulated datasets. This example uses two datasets simulated for this comparison.

The two datasets contain 100 individuals, each receiving 3 different doses:(0, 10, 90), (5, 25, 65) or (0, 20, 30). It is assumed that doses were given in a cross-over study with sufficient wash-out period to avoid a carry-over effect. Responses y_{ij} have been simulated with an $\mathrm{E}_{\max}$ model, a standard pharmacodynamic model:

$$y_{ij} = E_{0,i} \frac{D_{ij} E_{\mathrm{max,i}}}{D_{ij} + ED_{50,i}} + \epsilon_{ij} \tag{3.3}$$

For subject i:

- the regression variable is the dose received $x_{ij} = (D_i)$
- the vector of individual parameters is $\theta_i = (\ln(E_{0,i}), \ln(E_{\max,i}), \ln(ED_{50,i}))$
- the only available covariate is the gender w_i of the individual (0 for male and 1 for female)

The individual parameters were simulated assuming a log-normal distribution for all parameters, and a gender effect on $ED_{50,i}$:

$$\begin{aligned} \ln(E_{0,i}) &= \ln(E_0) + \eta_{i1} \\ \ln(E_{\max,i}) &= \ln(E_{\max}) + \eta_{i2} \\ \ln(ED_{50,i}) &= \ln(ED_{50}) + \beta w_i + \eta_{i3} \end{aligned} \tag{3.4}$$

In the simulations, the fixed effects were set to $(\ln(E_O), \ln(E_{\max}), \ln(ED_{50})) = (24, 100, 12)$. The covariance matrix of the random effects was a diagonal matrix. The variances of the random effects were (0.12 , 0.26 , 0.05). The residual variance was a constant variance, with $a^2 = 20$. The two data sets were simulated with different values of β:

- the first dataset was simulated with a gender effect, $\beta = 0.3$, and is available in the package under the name PD1.saem
- the second dataset was simulated under the null hypothesis, $\beta = 0$, and is available in the package under the name PD2.saem

The data is shown in figure 3.9.

The following code was used in R to run this example on the first dataset:

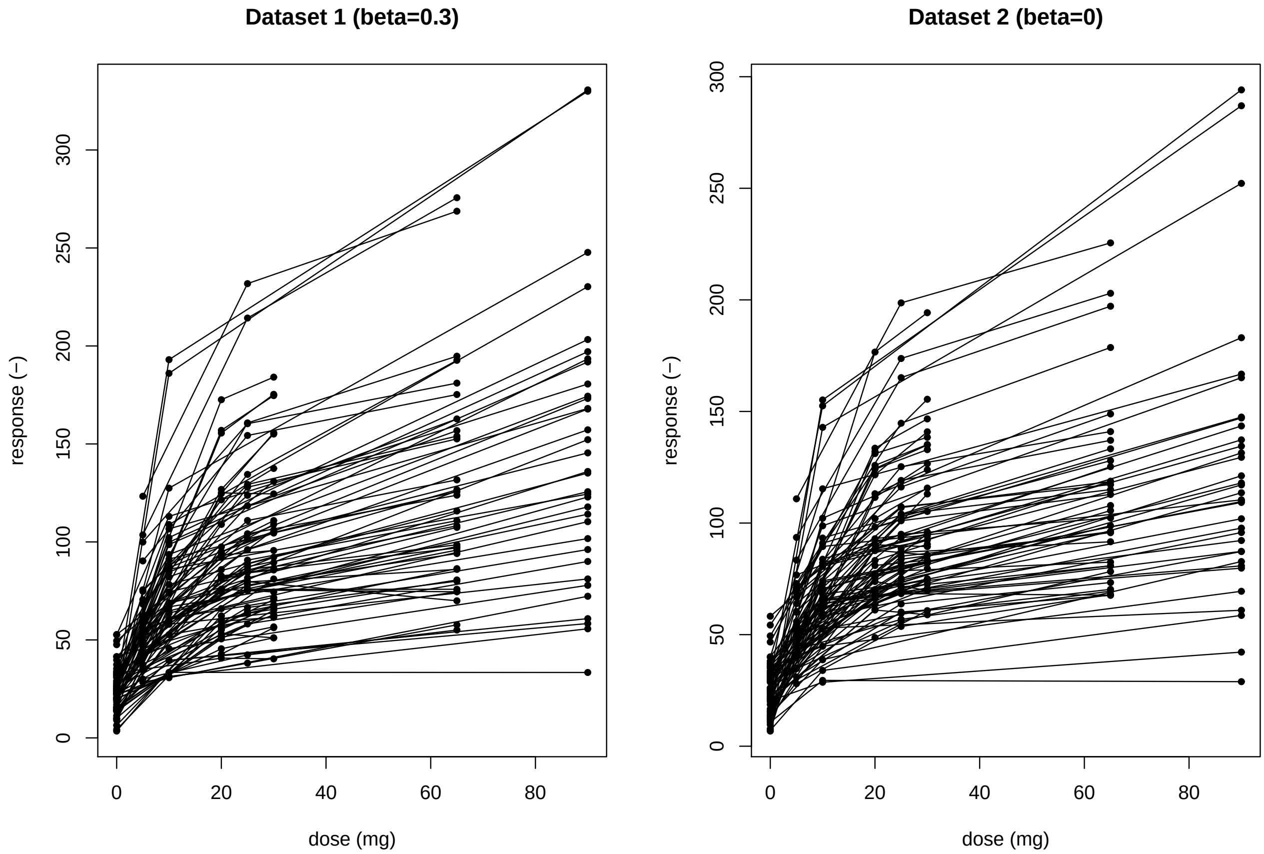

Figure 3.9: Effect versus dose for the data simulated with an E_{max} model, with a gender effect on ED_{50} (left) and without a gender effect (right).

```
library(saemix)
data(PD1.saemix)
data(PD2.saemix)
saemix.data1<-saemixData(name.data=PD1.saemix,header=TRUE,name.group=c("subject"),
name.predictors=c("dose"),name.response=c("response"),name.covariates=c("gender"),
units=list(x="mg",y="-",covariates="-"))

saemix.data2<-saemixData(name.data=PD2.saemix,header=TRUE,name.group=c("subject"),
name.predictors=c("dose"),name.response=c("response"),name.covariates=c("gender"),
units=list(x="mg",y="-",covariates="-"))

modelemax<-function(psi,id,xidep) {
# input:
#   psi : matrix of parameters (3 columns, E0, Emax, EC50)
#   id : vector of indices
```

```
#   xidep : dependent variables (same nb of rows as length of id)
# returns:
#   a vector of predictions of length equal to length of id
  dose<-xidep[,1]
  e0<-psi[id,1]
  emax<-psi[id,2]
  e50<-psi[id,3]
  f<-e0+emax*dose/(e50+dose)
  return(f)
}
saemix.model<-saemixModel(model=modelemax,description="Emax model",
psi0=matrix(c(20,300,20,0,0,0),ncol=3,byrow=TRUE,
dimnames=list(NULL,c("E0","Emax","EC50"))),transform.par=c(1,1,1),
covariate.model=matrix(c(0,0,1),ncol=3,byrow=TRUE),
fixed.estim=c(1,1,1),covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,
byrow=TRUE),error.model="constant")

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

# Fitting the model on the two PD datasets
saemix.fit1<-saemix(saemix.model,saemix.data1,saemix.options)
saemix.fit2<-saemix(saemix.model,saemix.data2,saemix.options)
```

Table 3.2 shows the parameter estimates for the two datasets. The estimates for the three fixed effects are similar for both datasets, while the estimate of β is close to the values simulated for both. For PD2.saemix, the SE on β is very large, as is the SE on the estimate of the variability of EC_{50}.

	PD1.saemix		PD2.saemix	
Parameter	Estimate (SE%)	IIV (SE%)	Estimate (SE%)	IIV (SE%)
E_0 (-)	22.71 (5%)	0.13 (22%)	23.18 (5%)	0.16 (20%)
$E_{\max}$ (-)	106.46 (6%)	0.31 (15%)	96.14 (5%)	0.22 (15%)
EC_{50} (mg)	11.25 (8%)	0.03 (55%)	12.38 (6%)	0.01 (151%)
$\beta_{gender,EC_{50}}$ (-)	0.35 (26%)	-	-0.06 (116%)	-
a (mg.L^{-1})	4.94 (8%)	-	4.67 (8%)	-

Table 3.2: Pharmacokinetic parameters estimated by SAEMIX for the simulated PD data.

The convergence plots are shown in figures 3.10 and 3.11, where we can see β converging to a non-zero value for the first dataset, while the estimate fluctuates around 0 for the second dataset. The Wald test performed for the fixed effect representing the effect of gender on EC_{50} shows that this parameter is significantly different from 0 in the first dataset (p=6.10^{-5}).

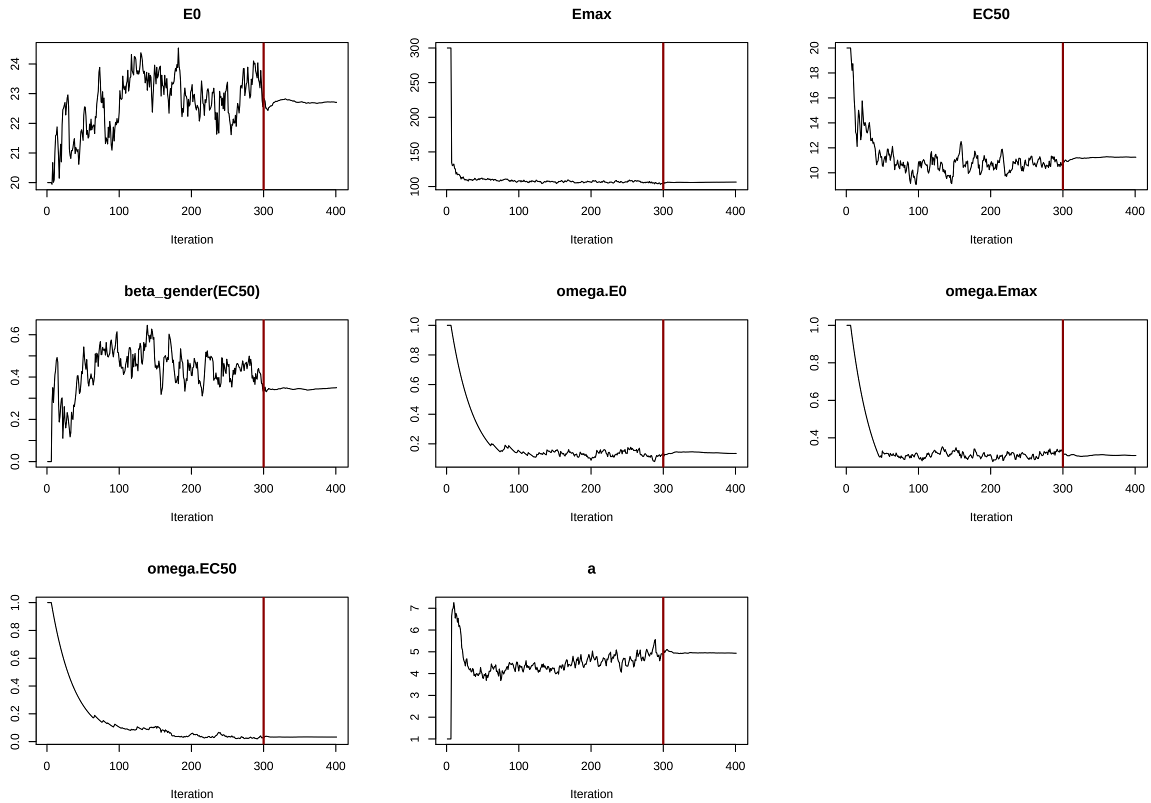

Figure 3.10: Convergence plots for the estimated pharmacokinetic parameters and the variabilities, for the first dataset.

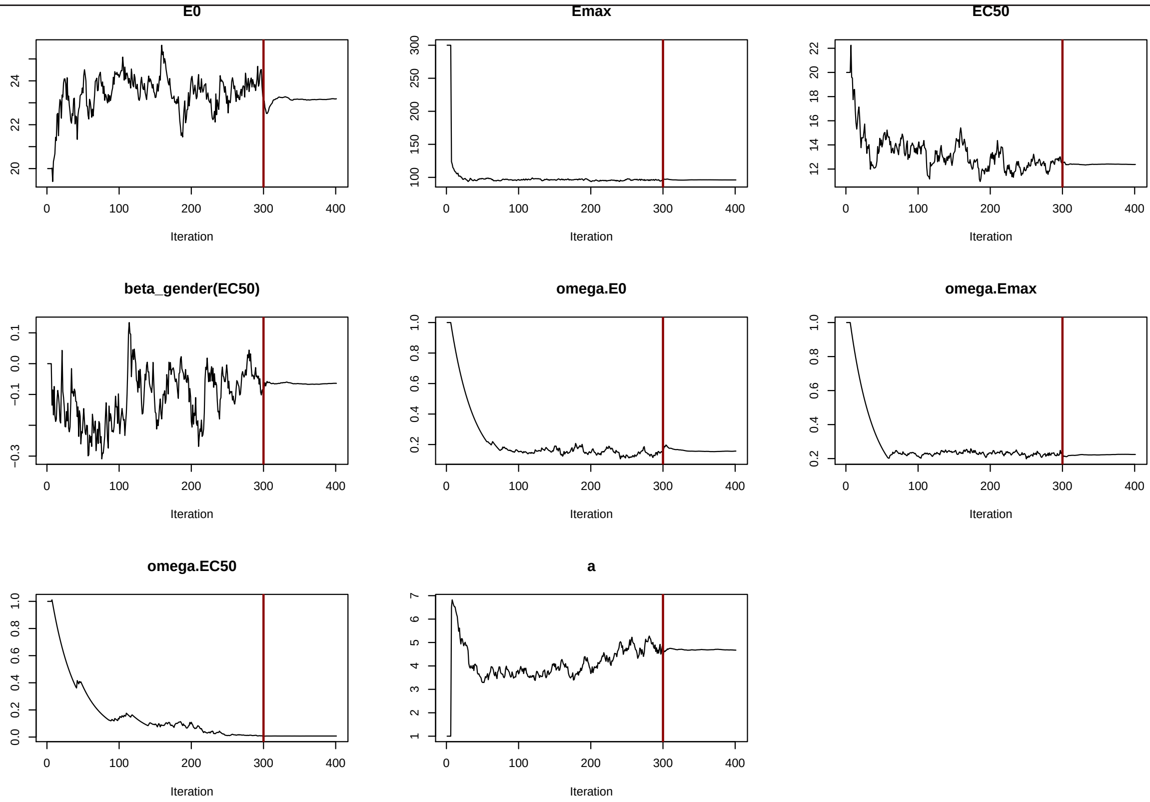

Figure 3.11: Convergence plots for the estimated pharmacokinetic parameters and the variabilities, for the second dataset.

Finally, figure 3.12 shows the individual data for the first 12 subjects in the first dataset, with the individual predictions overlayed. A smoothed prediction was obtained. The model fits the data extremely well, which is unsurprising given that this is simulated data, with a rather small residual variability.

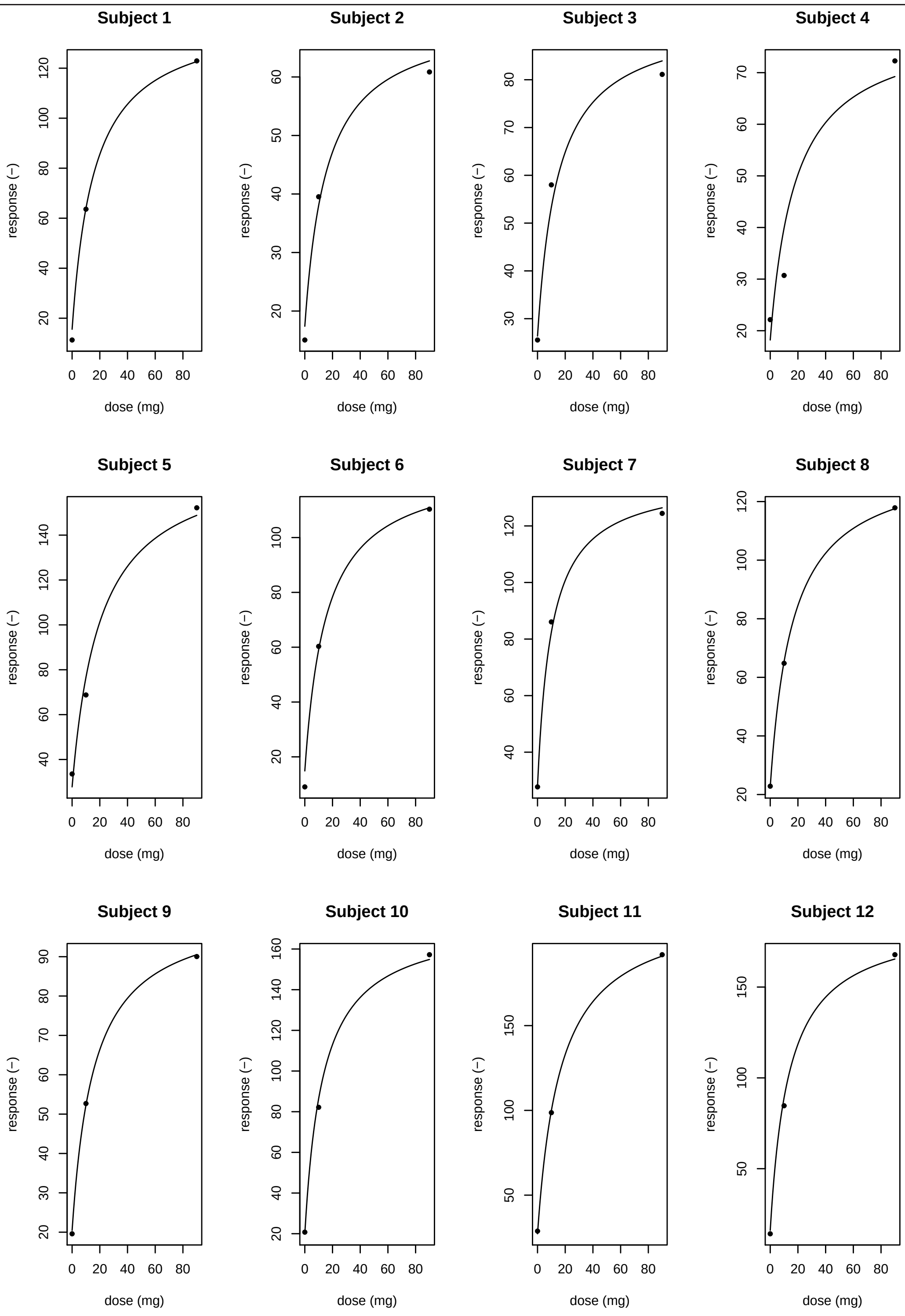

Figure 3.12: Individual plots for the 12 subjects in the first dataset (PD1.saemix). Dots represent observations and the line shows the profile predicted using the individual estimated parameters.

3.3 Weight gain of cows

The data used in this example is the evolution of the weight (in kg) of 560 cows. The weight of each cow was recorded on 9 or 10 occasions. An exponential model was assumed to describe the weight gain with time:

$$y_{ij} = A_i \ \left(1 - B_i e^{-K_i t_{ij}}\right) + \epsilon_{ij} \tag{3.5}$$

For subject i:

- the regression variable is the time (in days) $x_{ij} = (t_{ij})$
- the vector of individual parameters is $\theta_i = (A_i, B_i, K_i))$
- there were 3 covariates in the file:
  1. the year of birth (beetween 1988 and 1998)
  2. existence of a twin (no=1, yes=2)
  3. the rank of birth (beetween 3 and 7)

The data is shown in figure 3.13.

The following code was used in R to run this example:

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

growthcow<-function(psi,id,xidep) {
# input:
#   psi : matrix of parameters (3 columns, ka, V, CL)
#   id : vector of indices
#   xidep : dependent variables (same nb of rows as length of id)
# returns:
#   a vector of predictions of length equal to length of id
  x<-xidep[,1]
  a<-psi[id,1]
  b<-psi[id,2]
  k<-psi[id,3]
```

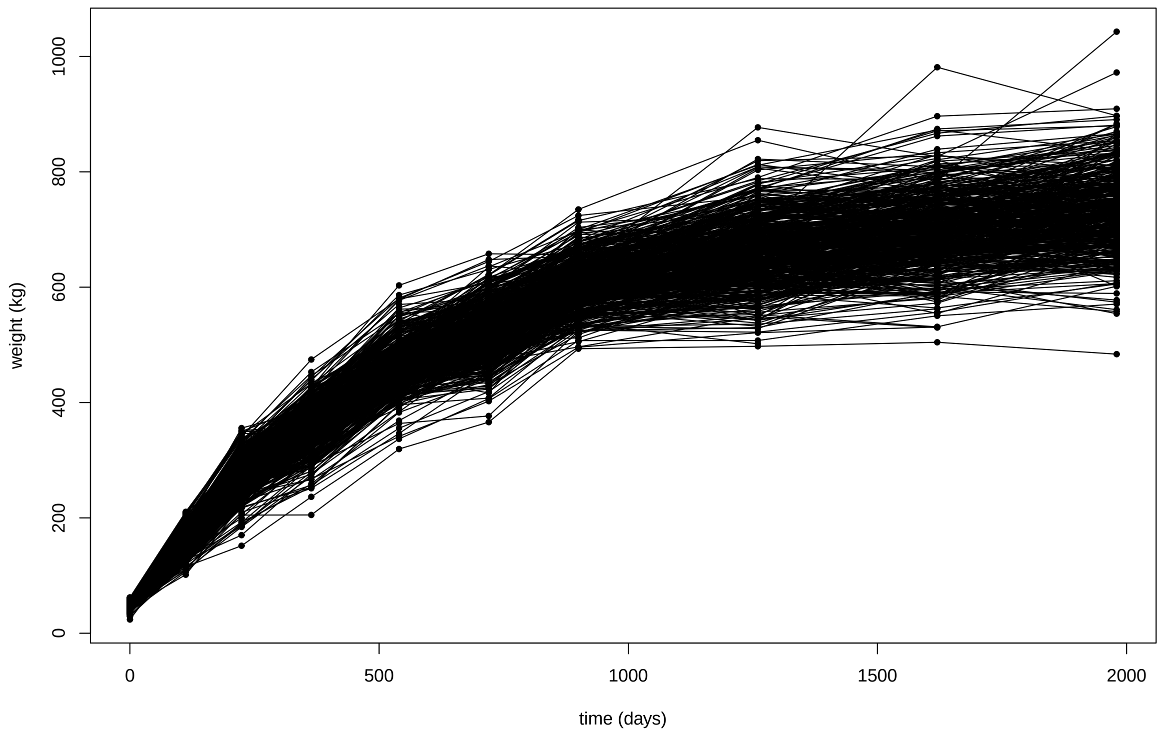

Figure 3.13: Weight gain of 560 cows recorded repeatedly over time.

```
  f<-a*(1-b*exp(-k*x))
  return(f)
}
saemix.model<-saemixModel(model=growthcow,description="Exponential model",
psi0=matrix(c(700,0.9,0.02,0,0,0),ncol=3,byrow=TRUE,
dimnames=list(NULL,c("A","B","k"))),transform.par=c(1,1,1),fixed.estim=c(1,1,1),
covariate.model=matrix(c(0,0,0,0,0,0,0,0,0),ncol=3,byrow=TRUE),
covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),
omega.init=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=TRUE),error.model="constant")

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

# Fitting the models
saemix.fit<-saemix(saemix.model,saemix.data,saemix.options)
```

As an alternative, we can compute the estimate of the likelihood by Gaussian Quadrature:

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

The three estimates of the likelihood were found to be in good agreement in this example:

```
----------------------------------------------------
---------------  Statistical criteria  -------------
----------------------------------------------------
Likelihood computed by linearisation
      -2LL= 53723.42
      AIC = 53737.42
      BIC = 53767.71

Likelihood computed by importance sampling
      -2LL= 53723.88
      AIC = 53737.88
      BIC = 53768.18

Likelihood computed by Gaussian quadrature
      -2LL= 53723.04
      AIC = 53737.04
      BIC = 53767.34
----------------------------------------------------
```

The fits to the data from the first 4 animals can be plotted using the function saemix.plot.fits. First, default plot options are set in a list called saemix.plot.options using the function saemix.plot.setoptions. Second, the option controlling the list of subjects to be plotted is set (here, we choose to plot the graphs for the first four animals), and the option smooth indicates that we want an smoothed version of the plots (using interpolated weights):

```
plot(saemix.fit,plot.type="individual.fit",ilist=1:4,smooth=TRUE)
```

The result is shown in figure 3.14.

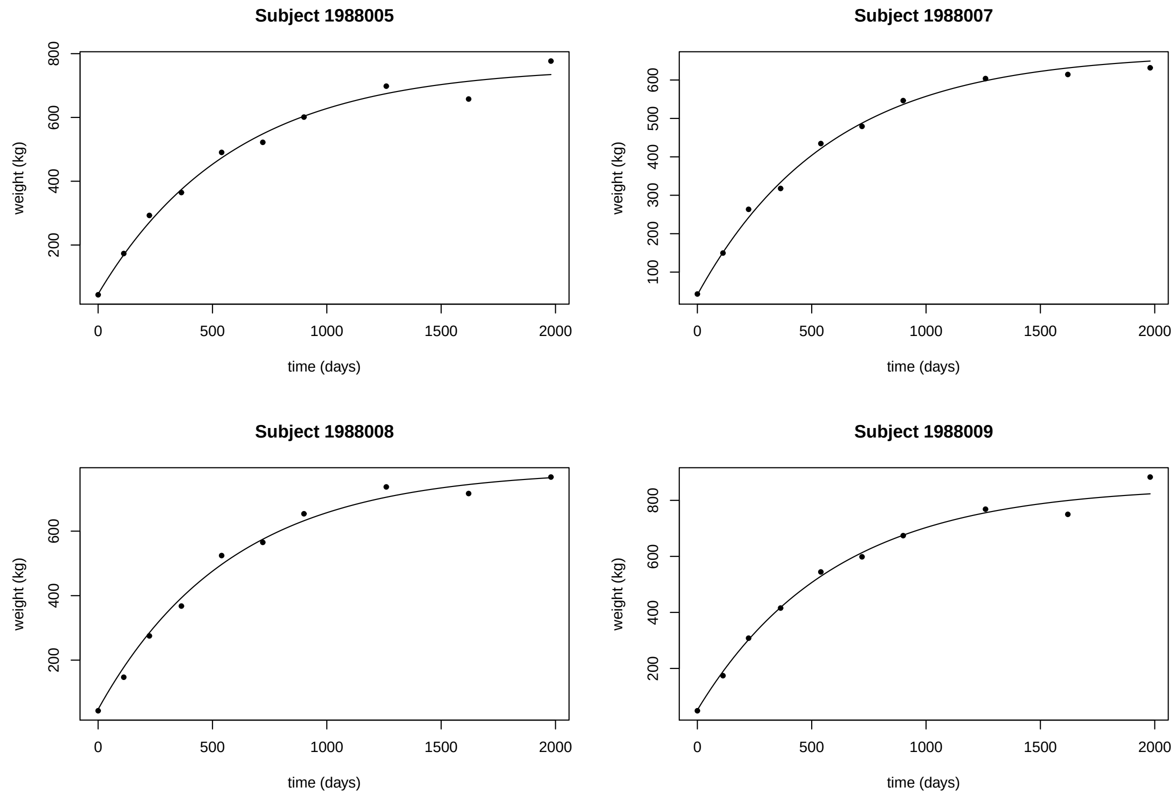

Figure 3.14: Fit for the first four subjects in the cow dataset.

3.4 Height of Oxford boys

SAEMIX can be used even for linear models. The dataset oxboys.saemix was taken from the library nlme [22]. It describes the evolution with age of the height of boys from Oxford, England. There is no covariate in the model, and we use a simple linear model to account for the increase in height over this age range:

$$y_{ij} = \mathrm{Base}_i + \mathrm{Slope}\ \mathrm{age}_{ij} + \epsilon_{ij} \tag{3.6}$$

where Base_i is the baseline height at the entrance of subject i in the study and Slope_i the slope for the increase of height with age age_{ij}. For subject i:

- the vector of regression (or design) variables is $x_{ij} = (\mathrm{age}_{ij})$
- the vector of individual parameters is $\theta_i = (\mathrm{Base}_i, \mathrm{Slope}_i)$
  - the individual parameters are assumed to have a normal distribution

- we can use a simple homoscedastic error model where Var $(\epsilon_{ij}) = a^2$

The data is shown in figure 3.15.

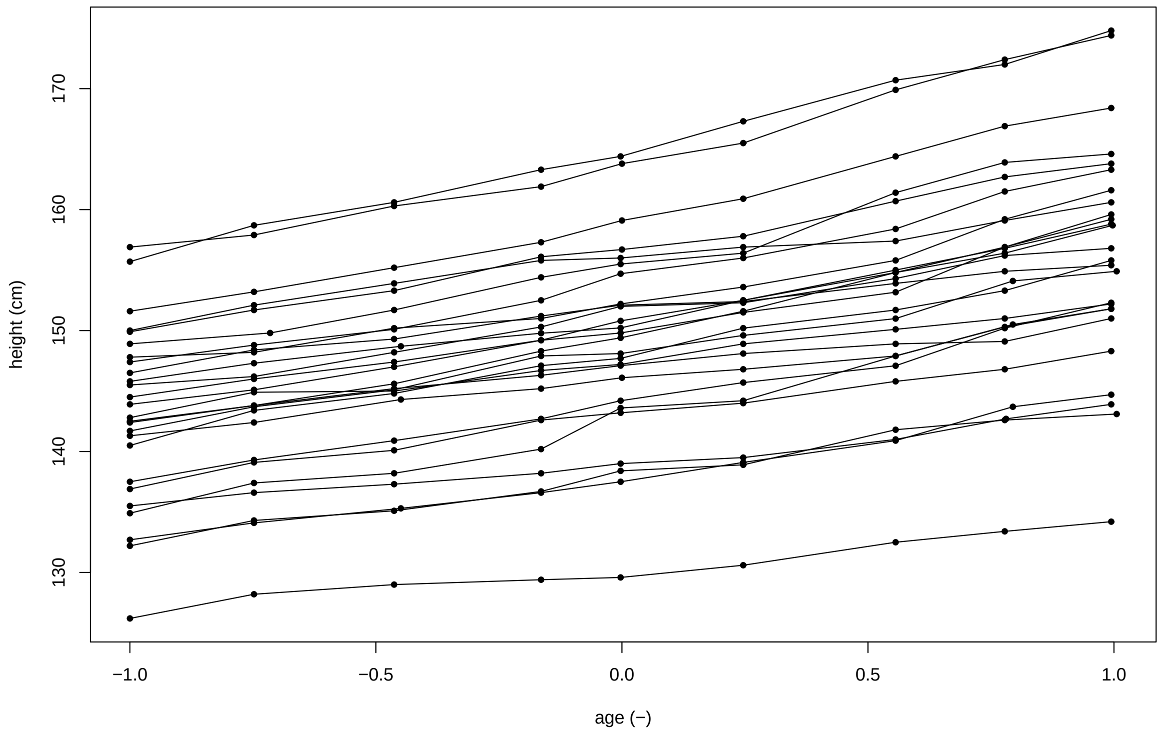

Figure 3.15: Evolution with age of the height of boys from Oxford.

The following code was used in R to run this example:

```
library(saemix)

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

growth.linear<-function(psi,id,xidep) {
# input:
#   psi : matrix of parameters (2 columns, base and slope)
#   id : vector of indices
#   xidep : dependent variables (same nb of rows as length of id)
# returns:
#   a vector of predictions of length equal to length of id
```

```
  x<-xidep[,1]
  base<-psi[id,1]
  slope<-psi[id,2]
  f<-base+slope*x
  return(f)
}
saemix.model<-saemixModel(model=growth.linear,description="Linear model",
psi0=matrix(c(140,1),ncol=2,byrow=T,dimnames=list(NULL,c("base","slope"))),
transform.par=c(1,0), covariance.model=matrix(c(1,1,1,1),ncol=2,byrow=T),
error.model="constant")

saemix.options<-list(algorithms=c(1,1,1),nb.chains=1)

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

3.5 A yield model

The data used in this study were from 37 winter wheat experiments carried out between 1990 and 1996 on commercial farms in the Paris Basin, France. Each experiment was from a different site. Two soil types were represented, a loam soil and a chalky soil. Common winter wheat varieties were used. Each experiment consisted of five to eight different nitrogen fertilizer rates, for a total of 224 nitrogen treatments. Nitrogen fertilizer was applied in two applications during the growing season. For each nitrogen treatment, grain yield (adjusted to 150 g.kg^{-1} grain moisture content) was measured. In addition, end-of-winter mineral soil nitrogen (NO3- plus NH4+) in the 0- to 90-cm layer was measured on each site-year during February when the crops were tillering. See [9] for a more complete description of the plant sampling and nitrogen analysis. Yield and end-of-winter mineral soil nitrogen measurements were in the ranges 3.44- 11.54 t.ha^{-1} , and 40-180 kg.ha^{-1} respectively.

The data is shown in figure 3.16.

Let y_{ij} denote the j^{th} measurement of the yield response in the i^{th} site-year when the nitrogen fertilizer dose d_{ij} is applied. The only available covariate is the amount of soil mineral nitrogen at the end of winter (w_i).

A first model is a linear-plus-plateau function (LP) defined by:

$$y_{ij} = \begin{cases} Y_{max,i} + B_i(d_{ij} - X_{max,i}) & \text{if } d_{ij} \leq X_{max,i} \\ Y_{max,i} & \text{if } d_{ij} \geq X_{max,i} \end{cases} \tag{3.7}$$

This model includes three individual random parameters, $\phi_i = (Y_{max,i}, X_{max,i}, B_i)$. $Y_{max,i}$ is the maximal yield value in the i^{th} site-year and $X_{max,i}$ is the fertilizer dose that maximizes yield. The three parameters were assumed to follow a normal distribution.

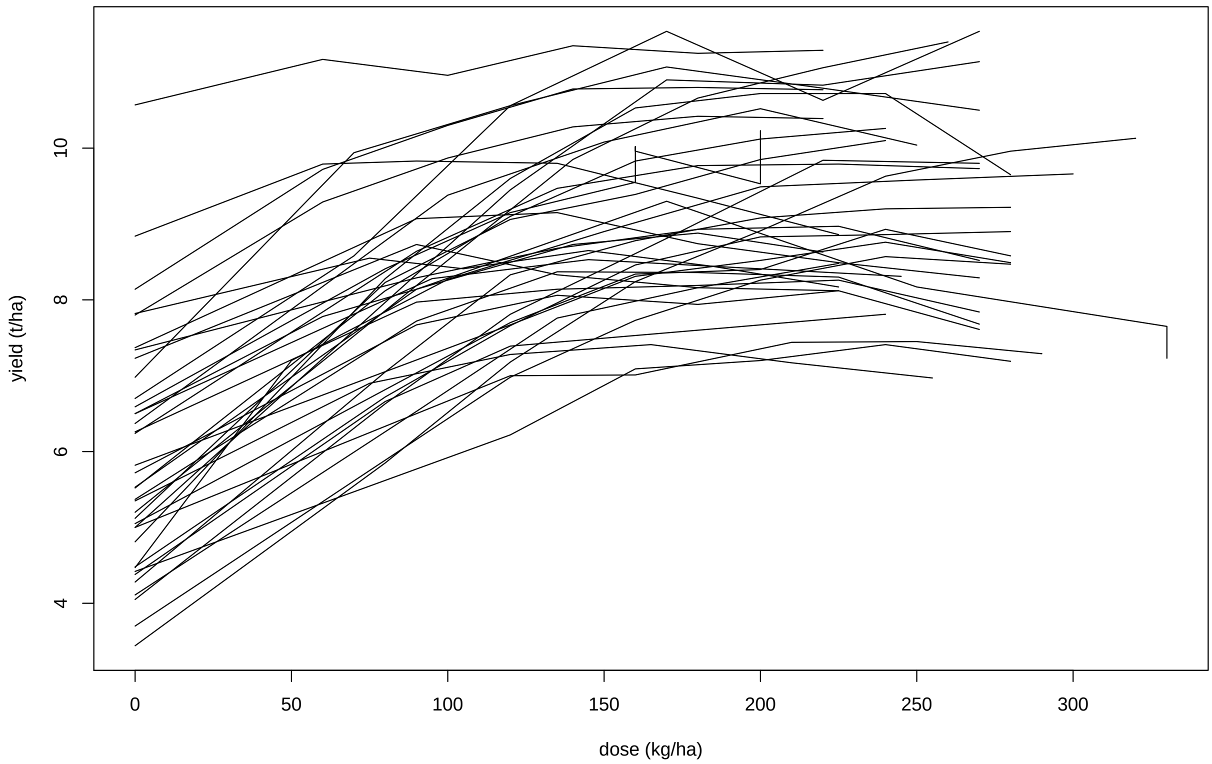

Figure 3.16: Yield from 37 winter wheat experiments.

A second model is a square-root-plus-plateau function (QP) defined by:

$$y_{ij} = \begin{cases} Y_{max,i} + B_i(\sqrt{d_{ij}} - \sqrt{X_{max,i}}) & \text{if } d_{ij} \leq X_{max,i} \\ Y_{max,i} & \text{if } d_{ij} \geq X_{max,i} \end{cases} \quad (3.8)$$

We use the following code to run these two models:

```
library(saemix)

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

yield.LP<-function(psi,id,xidep) {
  x<-xidep[,1]
  ymax<-psi[id,1]
```

```
  xmax<-psi[id,2]
  slope<-psi[id,3]
  f<-ymax+slope*(x-xmax)
#  cat(length(f),"  ",length(ymax),"\n")
  f[x>xmax]<-ymax[x>xmax]
  return(f)
}

yield.QP<-function(psi,id,xidep) {
  x<-xidep[,1]
  ymax<-psi[id,1]
  xmax<-psi[id,2]
  slope<-psi[id,3]
  f<-ymax+slope*(x**0.5-xmax**0.5)
#  f<-ymax+slope*sqrt(abs(x-xmax))
  f[x>xmax]<-ymax[x>xmax]
  return(f)
}
saemix.model1<-saemixModel(model=yield.LP,description="Linear + plateau model",
psi0=matrix(c(8,100,0.2,0,0,0),ncol=3,byrow=T, dimnames=list(NULL,c("Ymax","Xmax",
"slope"))), covariate.model=matrix(c(0,0,0),ncol=3,byrow=T),
transform.par=c(0,0,0),covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=T),
error.model="constant")

saemix.model2<-saemixModel(model=yield.QP,description="Quadratic + plateau model",
psi0=matrix(c(10,120,0.005,0,0,0),ncol=3,byrow=T, dimnames=list(NULL,c("Ymax","Xmax",
"slope"))), covariate.model=matrix(c(0,0,0),ncol=3,byrow=T), transform.par=c(0,0,0),
covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=T),error.model="constant")

saemix.options<-list(algorithms=c(1,1,1),nb.chains=1, nbiter.saemix=c(400,100),
nmc.is=25000, save=FALSE,save.graphs=FALSE)

# Fitting the models
saemix.fit1<-saemix(saemix.model1,saemix.data,saemix.options)
saemix.fit2<-saemix(saemix.model2,saemix.data,saemix.options)
```

The two models perform very similarly in terms of log-likelihood, with a slight advantage to the LP model: the statistical criterion (-2 times the log-likelihood) was equal to 406.86 for the LP model and to 416.28 for the QP model. Figure 3.17 shows the plots of predictions versus observations for the two models, again very similar.

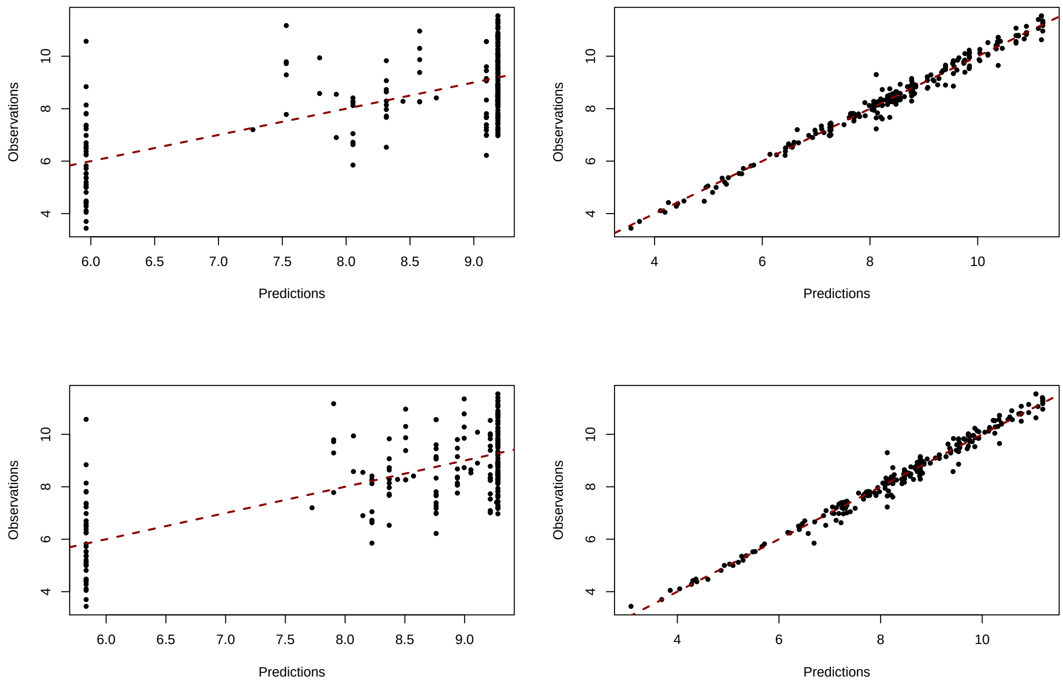

Figure 3.17: Observations versus predictions for the LP model (upper panel) and QP model (lower panel), with population predictions on the left and individual predictions on the right.

Figure 3.18 shows the fit of the two models for the first four subjects. The figure was obtained using the following code:

```
par(mfrow=c(4,2))
for(i in 1:4) {
  plot(saemix.fit1,plot.type="individual.fit",ilist=i,smooth=TRUE,new=F)
  plot(saemix.fit2,plot.type="individual.fit",ilist=i,smooth=TRUE,new=F)
}
```

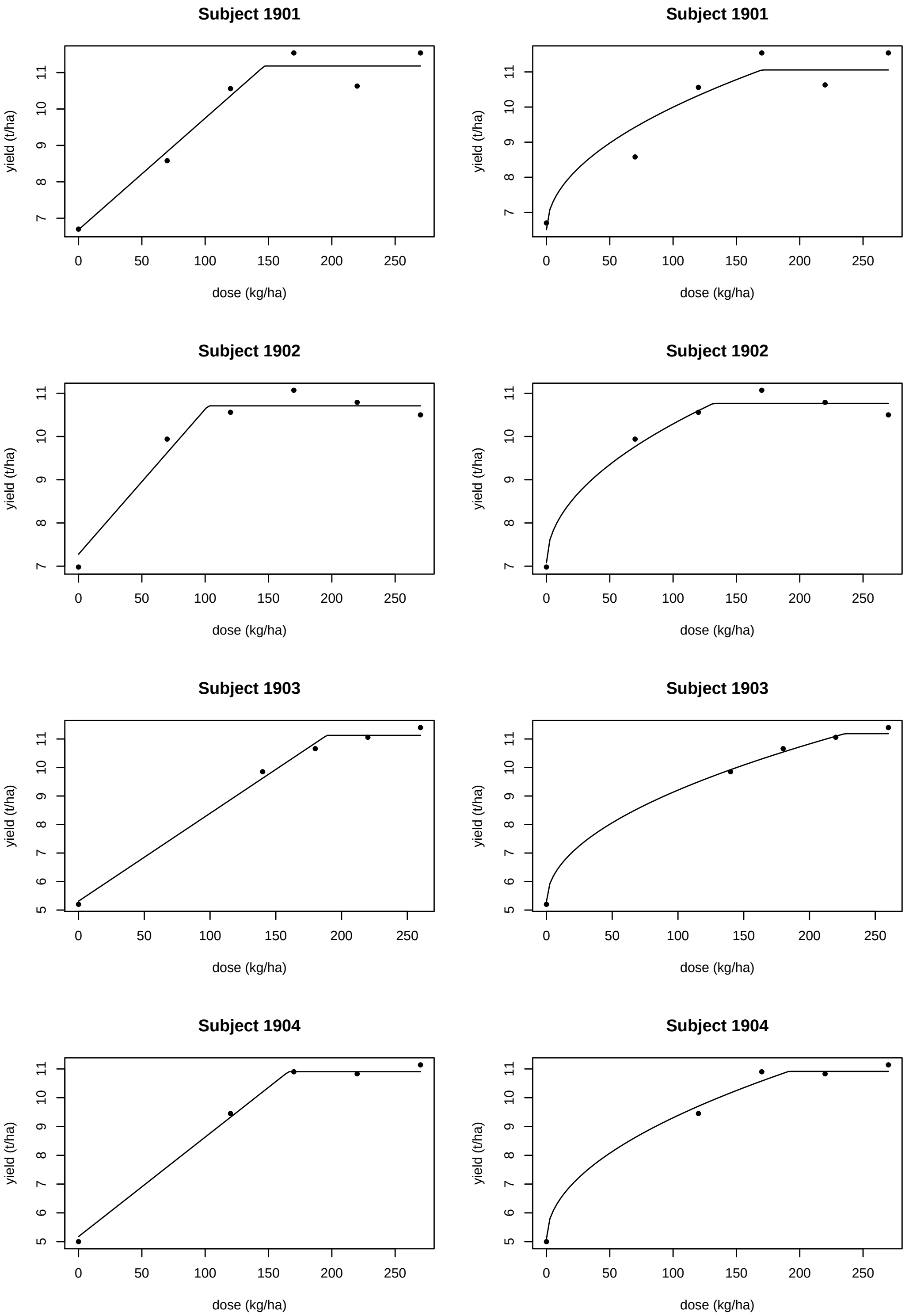

Figure 3.18: Fits for the LP model (left) and QP model (right) for the first 4 subjects.

We can explore the covariates using diagnostic plots. For instance, the following code plots the estimated individual parameters versus the covariates in the model (here, soil nitrogen), assuming the fit is in the object saemix.fit:

```
plot(saemix.fit1, plot.type="parameters.vs.covariates")
```

Figure 3.19 shows the result, and indicates a decreasing trend in X_{max} with increasing amounts of soil nitrogen.

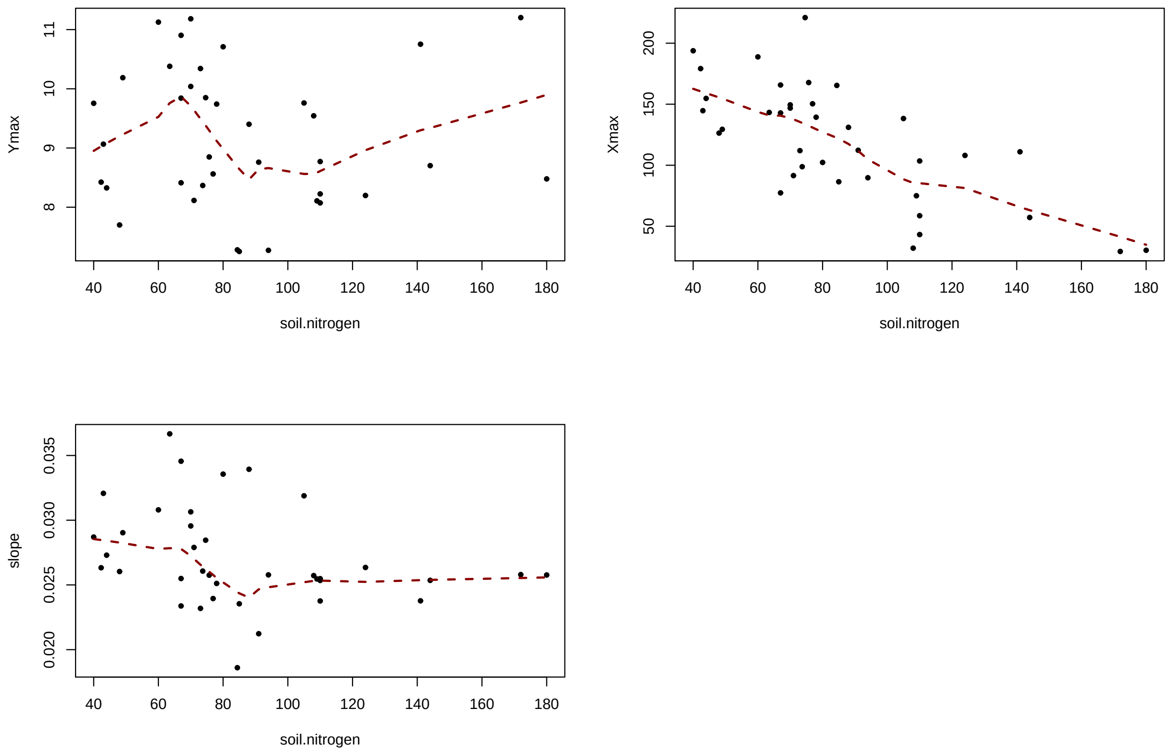

Figure 3.19: Graphs of the estimated (individual) parameters versus covariate.

In this example, we can then show with SAEMIX that the effect of the amount of soil mineral nitrogen at the end of winter is statistically significant for explaining the fluctuations of the parameter X_{max} in both LP and QP models, with a drop of over 20 points in the statistical criterion. For example, the following code shows how to fit the LP-model with and without covariate effect on $\mathrm{X}_{\mathrm{max}}$, and outputs the resulting log-likelihoods:

```
saemix.model3<-saemixModel(model=yield.LP,description="Linear + plateau model",
psi0=matrix(c(8,100,0.2,0,0,0),ncol=3,byrow=T, dimnames=list(NULL,c("Ymax","Xmax",
```

```
"slope"))), covariate.model=matrix(c(0,1,0),ncol=3,byrow=T),
transform.par=c(0,0,0),covariance.model=matrix(c(1,0,0,0,1,0,0,0,1),ncol=3,byrow=T),
error.model="constant")

saemix.fit3<-saemix(saemix.model3,saemix.data,saemix.options)

{
  cat("LP model:\n")
  cat("     without covariate, -2xLL=",(saemix.fit1["results"]["ll.is"])*(-2),"\n")
  cat("     with covariate, -2xLL=",(saemix.fit3["results"]["ll.is"])*(-2),"\n")
}

# The same can be done for the QP model
```

Chapter 4

S4 classes

This section is in progress. More information on S4 classes and Rpackages can be found in tutorials on the Web. I used extensively the following manual [9].

4.1 A very short introduction to S4 classes

SAEMIX has been programmed using the S4 classes in R. S4 classes implement Object oriented programming (OOP) in R, allowing to construct modular pieces of code which can be used as black boxes for large systems. Most packages in the base library and many contributed packages use the former class system, called S3. However, S4 classes are a more traditional and complete object oriented system including type checking and multiple dispatching. S4 is implemented in the methods package in base R.

Elements of an object are called "Slots". Slots can be accessed using the @ operator, instead of the $ operator used for lists. However, the use of @ to access the class slots is heavily discouraged outside functions programmed directly by package developers. Instead, in SAEMIX accessor functions ("get" functions) and replacement functions ("set" functions) have been defined to allow access to elements of an object in the same way as one would the elements of a list in R, through the name of the slot. If obj is an object with a slot called "slot", we would display the value of the slot using the command:

```
obj["slot"]
```

Assuming that slot is a character string, we can replace its value by "my string" using the command:

```
obj["slot"]<-"my string"
```

Since an S4 class has built-in check for types, a command such as:

```
obj["slot"]<-3
```

would in this case produce an error.

4.2 S4 classes used in SAEMIX

4.2.1 S4 objects

Visible S4 objects

The following classes have been defined in SAEMIX:

- SaemixData: this object contains the structure and data of the longitudinal dataset
- SaemixModel: this object contains the structure representing a non-linear mixed effect model, used by the SAEM algorithm
- SaemixRes: this object contains the results obtained after a fit by saemix(); it is included in the SaemixObject as the Slot results
- SaemixObject: this is the object returned by a call to saemix(); this object has the following slots:
  - data: a SaemixData object, containing the structure and data of the longitudinal dataset
  - model: a SaemixModel object, containing the characteristics of the non-linear mixed effect model
  - results: a SaemixRes object, containing the results obtained after a fit by saemix()
  - options: a list of options
  - prefs: a list of graphical preferences, that will replace the default graphical preferences if changed; the preferences set in this list can be superseded by setting an option in the call to the plot functions (see section 2.7.3)
  - rep.data: an object of class SaemixRepData produced during the fit of the SAEM algorithm (for internal use only)
  - sim.data: an object of class SaemixSimData containing data simulated according to the design of the original dataset and the fitted model, with the results obtained during the fit

The constructor functions for the first two objects are respectively saemixData() and saemix-Model() (with a lowercase initial letter, to distinguish it from the object classes, which start with a capital letter, since in Rlowercase and uppercase letters are different). These two functions are the functions intended to be used directly to produce the objects given as input to the saemix() function.

- saemixData(): the saemixData() function requires one mandatory argument, the name of a dataframe in Ror of a file on disk containing the data. If the file has a header (or if the dataframe has column names), the program will attempt to recognise suitable names for the grouping, predictor and response variables. These may also be specified by the user, either as names or column numbers (see help page for SaemixData).
- saemixModel(): the saemixModel() function requires two mandatory arguments: the name of a Rfunction computing the model in the SAEMIX format (see details and examples) and a matrix giving the initial estimates of the fixed parameters in the model. This matrix should contain at least one row, with the values of the initial estimates for the population mean parameters; if covariates are present in the data and enter the model, a second row should contain the values of the initial estimates for the covariate effects.

There is no constructor function for an SaemixObject object, since such an object should be returned by the saemix() function.

Hidden S4 objects

In addition to the visible objects, SAEMIX also has 2 other classes which are not intended to be used directly by the user:

- SaemixRepData: this object is created during the fit by saemix()
- SaemixSimData: this object is created when simulating data

An SaemixObject contains instances of these two classes. The slot of class SaemixRepData is produced and filled during the fit, while the slot of class SaemixSimData is produced when simulations are performed (in particular, to compute weighted residuals and npde, and produce VPC plots).

4.2.2 Methods for S4 objects in SAEMIX

Two types of functions have been developed for the SAEMIX package:

- methods

- classical functions

Methods are a special type of functions, which apply to objects and benefit from multiple dispatch. Ruses multiple dispatch extensively: one generic function call, such as for instance print, is capable of dispatching on the type of its argument and calls a printing function that is specific to the data supplied. For instance, using the print() function on a matrix will output the matrix, while using the same function on an object returned by the lm() function will produce a summary of the linear regression fit. We used this feature to produce notably plot() and print() functions (see next sections) which should apply to our SAEMIX objects in a user-friendly way.

Generic methods

The following generic methods have been defined for SaemixData, SaemixModel and SaemixObject objects:

- print: the print function produces a summary of the object in a nice format
- show: this function is used invisibly by Rwhen the name of the object is typed, and produces a short summary of the object
- summary: this function produces a summary of the object, and invisibly returns a list with a number of elements, which provides an alternative way to access elements of the class
  - for SaemixData, the list contains ntot (total number of observations), nind (vector containing the number of observations for each subject), id (vector of identifier), xind (matrix of predictors), cov (matrix of individual covariates), y (observations);
  - for SaemixModel, the list contains the model function, the error model, the list of parameters, the covariance structure, the covariate model;
  - for SaemixObject, the list contains the estimated fixed effects, the estimated parameters of the residual error model, the estimated variability of the random effects, the correlation matrix, the log-likelihood by the different methods used, the Fisher information matrix, the population and individual estimates of the parameters for each subject, the fitted values, the residuals.
- plot: this produces plots of the different objects
  - for SaemixData, a plot of the data is produced. The default plot is a spaghetti plot of the response variable versus the predictor (if several predictors, this is the predictor given by name.X) with a different line for each individual
  - for SaemixModel, the model is used to predict the value of the response variable according to the value of the predictor(s) over a given range of values for the main predictor.

  – for SaemixObject, the plot function produces a number of different plots. By default, a series of plot are produced; when called with the plot.type argument, selected plots can be chosen.

- [function: the get function, used to access the value of the slots in an object
- [<-: function: the set function, used to replace the value of the slots in an object

Examples of calls to these functions are given in the corresponding man pages and in the documentation (chapter 3). Additional generic methods for classes, such as initialize(), are not user-level in the SAEMIX package.

Specific methods

Specific methods have been developed for the objects in the SAEMIX package. Specific methods are methods which possibly apply to objects of several classes. For all purposes, they are used like generic methods.

The following methods apply to SaemixObject objects:

- showall: this method produces an extensive summary of the object. This method is also defined for SaemixData and SaemixModel objects.
- predict: this function uses the results from an SAEM fit to obtain model predictions for the data in the data element of the SaemixObject object
- psi, phi, eta: these three methods are used to access the estimates of the individual parameters and random effects. When the object passed to the function does not contain these estimates, they are automatically computed. The object is then returned (invisibly) with these estimates added to the results
- coef: this method extracts the coefficients from an SaemixObject fit, returning a list with three components (some components may be empty (eg MAP estimates) if they have not been computed during or after the fit)
  – fixed: estimated fixed effects in the model
  – population: population parameter estimates for each subject; the estimation of population parameters includes individual covariates if some enter ther model; this is a list with two components, map and cond, which are respectively the MAP estimates and the conditional mean estimates
  – individual: individual parameter estimates: a list with two components map and cond; this is a list with two components, map and cond, which are respectively the MAP estimates and the conditional mean estimates of the individual parameters

Additional specific methods have been defined but are not user-level (read.saemixData() is used by the constructor function).

4.2.3 Accessing S4 objects in SAEMIX

Help for S4 objects and methods

Aliases for the SaemixData, SaemixModel and SaemixObject objects have been created, so that the usual online help can be called:

```
help(SaemixData)
?SaemixData
```

Classic methods are accessed by the usual help function, for example:

```
?saemix
```

will produce the help file for the main saemix() function, fitting the non-linear mixed effect model.

The help files for generic and methods on the other hand can be accessed by the following (non-intuitive) commands:

```
help("plot,SaemixData")
```

Typing:

```
help(plot)
```

will only give the help page for the generic Rplot function. In the same way, we access the help page for the plot function applied to the object resulting from a call to saemix() (which contains links to the page describing the specific plots):

```
help("plot,SaemixObject")
```

Elements for S4 objects defined in SAEMIX

The elements, or slots, of the objects with class SaemixData, SaemixModel and SaemixObject are described in the respective help pages. When an object is first created, some of its slots may be empty or filled in with default values.

In the following, we create the object saemix.data by a call to the constructor function:

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

We can then access the number of subjects in the dataset by the get function:

```
saemix.data["N"]
```

Warning: modifying the elements in the objects outside of dedicated functions or methods can have unwanted side-effects. For instance, if one was to change the number of subjects in the data slot of an object created by a call to saemix(), the consistency of the object would not be guaranteed, and this could cause strange behaviour when trying to print or plot the object, or use it in subsequent functions. For this reason it is strongly recommended to only use the functions and methods defined in SAEMIX to access and modify SAEMIX objects. For instance, to apply the SAEM algorithm only to a subset of the subjects, it is preferable to apply the function saemixData() to a subset of the data instead of trying the change directly the SaemixData object.

Bibliography

[1] Bertrand, J., Comets, E., Laffont, C., Chenel, M., and Mentré, F. Pharmacogenetics and population pharmacokinetics: impact of the design on three tests using the SAEM algorithm. *Journal of Pharmacokinetics and Pharmacodynamics 36* (2009), 317–39.

[2] Comets, E., Brendel, K., and Mentré, F. 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* (2008), 154–66.

[3] Comets, E., Verstuyft, C., Lavielle, M., Jaillon, P., Becquemont, L., and Mentre, F. Modelling the influence of MDR1 polymorphism on digoxin pharmacokinetic parameters. *European Journal of Clinical Pharmacology 63* (2007), 437–449.

[4] Davidian, M., and Giltinan, D. *Nonlinear models for repeated measurement data*. Chapman & Hall, London, 1995.

[5] Davidian, M., and Giltinan, D. Nonlinear models for repeated measurements: An overview and update. *JABES 8* (2003), 387–419.

[6] Delyon, B., Lavielle, M., and Moulines, E. Convergence of a stochastic approximation version of the EM algorithm. *Annals of Statistics 27* (1999), 94–128.

[7] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. Ser. B 39*, 1 (1977), 1–38. With discussion.

[8] Donnet, S., and Samson, A. Estimation of parameters in incomplete data models defined by dynamical systems. *Journ. of Stat. and Plan. Infer. 50* (2007), 2381–2398.

[9] Genolini, C. *Construire un Package - Classic et S4*. INSERM U669, Paris, France, 2010.

[10] Girard, P., and Mentré, F. A comparison of estimation methods in nonlinear mixed effects models using a blind analysis (oral presentation). *PAGE, Pamplona* (2005).

[11] Jaffrézic, F., Meza, C., Foulley, J., and Lavielle, M. The SAEM algorithm for the analysis of nonlinear traits in genetic studies. *Genetics Selection Evolution 38* (2006), 583–600.

[12] Kuhn, E., and Lavielle, M. Coupling a stochastic approximation version of EM with a MCMC procedure. *ESAIM P&S 8* (2004), 115–131.

[13] Kuhn, E., and Lavielle, M. Maximum likelihood estimation in nonlinear mixed effects models. *Computational Statistics and Data Analysis 49* (2005), 1020–1038.

[14] Lavielle, M. *MONOLIX (MOdèles NOn LInéaires à effets miXtes)*. MONOLIX group, Orsay, France, 2005.

[15] Lavielle, M., and Kuhn, E. Maximum likelihood estimation in nonlinear mixed effects models (oral communication). *PAGE, Verona* (2003).

[16] Lavielle, M., and Mentré, F. Estimation of population pharmacokinetic parameters of saquinavir in HIV patients and covariate analysis with MONOLIX (poster). *PAGE, Pamplona* (2005).

[17] Lavielle, M., and Mentré, F. Estimation of population pharmacokinetic parameters of saquinavir in HIV patients with the MONOLIX software. *Journal of Pharmacokinetics and Pharmacodynamics 34*, 2 (2007), 229–249.

[18] Louis, T. A. Finding the observed information matrix when using the EM algorithm. *J. Roy. Statist. Soc. Ser. B 44*, 2 (1982), 226–233.

[19] Makowski, D., and Lavielle, M. Using SAEM to estimate parameters of models of response to applied fertilizer. *Jour. of Agr., Bio, and Env. Stat. 11*, 1 (2006), 45–60.

[20] Panhard, X., and Samson, A. Extension of the SAEM algorithm for the estimation of inter-occasion variability: application to the population pharmacokinetics of nelfinavir and its metabolite m8 (poster). *PAGE, Brugge* (2006).

[21] Pinheiro, J., and Bates, D. Approximations to the log-likelihood function in the non-linear mixed-effect models. *Journal of Computational and Graphical Statistics 4* (1995), 12–35.

[22] Pinheiro, J., Bates, D., DebRoy, S., Sarkar, D., and the R Core team. *nlme: Linear and Nonlinear Mixed Effects Models*, 2009. R package version 3.1-96.

[23] Pinheiro, J. C., and Bates, D. M. *Mixed-Effects Models in S and S-PLUS*. Springer, New York, 2000.

[24] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006. ISBN 3-900051-07-0.

[25] Raftery, A. Bayesian model selection in social research (with discussion). *Sociol Methodol* (1995), 111–95.

[26] Samson, A., Lavielle, M., and Mentré, F. Approximation EM algorithm in nonlinear mixed effects models: an evaluation by simulation (oral communication). *PAGE, Uppsala* (2004).

[27] Samson, A., Lavielle, M., and Mentré, F. Extension of the SAEM algorithm to left-censored data in nonlinear mixed-effects model: application to HIV dynamics model. *Computational Statistics and Data Analysis 51* (2006), 1562–1574.

[28] Samson, A., Lavielle, M., and Mentré, F. The SAEM algorithm for non-linear mixed models with left-censored data and differential systems: application to the joint modeling of hiv viral load and cd4 dynamics under treatment (oral presentation). *PAGE, Brugge* (2006).

[29] Samson, A., Lavielle, M., and Mentré, F. The SAEM algorithm for group comparison tests in longitudinal data analysis based on nonlinear mixed-effects model. *Stat. in Med. 26* (2007), 4860–4875.

[30] Samson, A., Panhard, X., Lavielle, M., and Mentré, F. Generalisation of the SAEM algorithm to nonlinear mixed effects model defined by differential equations: application to HIV viral dynamic models (poster). *PAGE, Pamplona* (2005).

[31] Sheiner, L., and Beal, S. *NONMEM Version 5.1.* University of California, NONMEM Project Group, San Francisco, 1998.

[32] Wu, C.-F. J. On the convergence properties of the EM algorithm. *Ann. Statist. 11*, 1 (1983), 95–103.