

REMixed : Regularized Estimation for non-linear Mixed effects models in the presence of large-dimension biomarkers in R

Auriane Gabaut^{1,2,3}, Cécile Proust-Lima^{1†}, Mélanie Prague^{1,2,3†}

¹ : Univ. Bordeaux, INSERM, BPH, U1219, Bordeaux, France

² : Univ. Bordeaux, Inria, Bordeaux, France

³ : Vaccine Research Institute, Paris, France

* : melanie.prague@inria.fr

† : These authors contributed equally

Abstract

The R package **REMixed** (Regularized Estimation for non-linear Mixed Effects Models) implements an iterative algorithm to estimate population parameters in non-linear mixed effects models (NLMEM) based on a structural mechanistic system and simultaneously regularizes observed biomarkers linked linearly to latent compartments of the mechanistic model. The algorithm uses a cyclical coordinate descent approach, alternating between regularization of the biomarker-compartment parameters through Lasso-penalized log-likelihood maximization and estimation of the remaining parameters through maximum likelihood estimation. The package relies on the Monolix software, which implements estimation methods for NLMEMs based on ordinary differential equations (ODEs). The package is illustrated on a simulated dataset of the humoral response after Ebola virus vaccination strategy using transcriptomics and viral load.

Contents

1	Introduction	2
2	REMixed algorithm	3
2.1	ODE-based NLMEMs	3
2.1.1	Model definition	3
2.1.2	Log-likelihood of the model	4
2.2	Overview of the estimation process	5
2.3	Implementation details	7
2.3.1	Regularization step	7
2.3.2	Log-likelihood estimation	8
2.3.2.1	Log-likelihood derivatives	8
2.3.2.2	Log-likelihood approximation by Gauss–Hermite quadrature	9
2.3.3	NLME parameters estimation	10
2.3.4	Selection of the penalty parameter	10
2.3.5	Final tests	11
2.3.6	Initialization strategy	11

3	Project architecture and connection with Monolix	12
3.1	Monolix project	12
3.2	Summary files and temporary projects	13
4	Usage	13
4.1	Demo project overview	13
4.2	Initialization	18
4.3	Grid Search with REMixed	19
4.4	REMixed with fixed penalty parameter λ	21
4.5	Final Statistical Tests	22
5	Conclusion	25
6	Appendix	29
A	Regularization step details	29
B	Structure of the project	32
C	Mlxtran model file for demo monolix project	33
D	Summary file for initialization task	34
E	Summary file for grid search	36
F	Summary file for REMix algorithm	43
G	Summary file for final tests and estimation	50

1 Introduction

Mechanistic models based on ordinary differential equations (ODEs) are widely used to describe biological systems, particularly in fields such as pharmacometrics, immunology, and epidemic modeling (Ibarra et al., 2021; Perelson and Ribeiro, 2018; Ganser et al., 2024). These models capture the temporal dynamics of multiple compartments, both observed and unobserved, involved in biological processes such as immune responses. However, achieving identifiability often requires repeated measurements across all compartments, which is typically not feasible in clinical studies (Lavielle, 2014).

In parallel, diverse longitudinal data such as transcriptomic profiles, increasingly accessible through high-throughput technologies (Lowe et al., 2017), provide indirect but rich information on underlying biological processes. This has led to a growing interest in incorporating large-dimensional longitudinal markers into statistical models to improve the estimation of such mechanisms, with encouraging results reported in semi-parametric frameworks, mixed-effects models, and joint models with survival outcomes (Ni et al., 2010; Zhao et al., 2014; Pan and Huang, 2014; Li et al., 2018; Sun and Basu, 2024). Many of these approaches rely on iterative estimation and variable selection strategies, demonstrating robustness in complex settings (Pan and Huang, 2014; Li et al., 2018; Caillebotte et al., 2024). Previous work has also demonstrated the feasibility of integrating transcriptomic data into non-linear mixed-effects models (NLMEMs). However, existing contributions remain limited to time-fixed covariates and do not capture dynamics through ODEs (Caillebotte et al.,

2024; Gabaut et al., 2025). The REMixed algorithm proposes a new approach that combines the estimation of ODE-based NLMEMs with the selection of time-varying longitudinal biomarkers linearly linked to one latent compartment of the ODE system.

We developed the R package **REMixed**, available on CRAN, to provide a flexible implementation of this methodology. The package offers tools for estimating ODE-based NLMEMs with longitudinal outcomes while simultaneously selecting transcriptomic markers dynamically associated with one latent biological state from the same mechanistic model. It relies on the **Monolix software** (Lixoft SAS, a Simulations Plus company, 2023a), which is accessible in R via the **lixoftConnectors** package and is freely available for academic use.

The proposed algorithm is an iterative procedure designed to solve Lasso-penalized maximum likelihood estimation (Tibshirani, 1996), implemented via cyclical coordinate descent (Tseng, 2001; Friedman et al., 2010). This strategy, previously applied in large-dimensional joint models for survival and mixed effect (Caillebotte et al., 2024), as well as for the selection and estimation fixed or random effects (Pan and Huang, 2014; Li et al., 2018), has shown strong performance in similar frameworks. The algorithm alternates between two steps: a regularization step and a mechanistic inference step. In the regularization step, coefficients linking biomarkers to latent compartments are updated by solving the first-order optimality conditions of the penalized log-likelihood, using second-order Taylor approximations. In the mechanistic inference step, model parameters are estimated with the Stochastic Approximation Expectation-Maximization (SAEM) algorithm (Dempster et al., 1977; Delyon et al., 1999), as implemented in Monolix (Kuhn and Lavielle, 2005; Lixoft SAS, a Simulations Plus company, 2023a).

In this vignette, we first describe the REMixed algorithm in Section 2. We then outline its implementation in the **REMixed** package and present the code structure and its integration with Monolix in Section 3. Finally, we illustrate the use of the main functions on a simulated dataset of humoral immune response following Ebola vaccination (Pasin et al., 2019) in Section 4.

2 REMixed algorithm

2.1 ODE-based NLMEMs

2.1.1 Model definition

Let us consider a mechanistic model defined by a system of ODEs involving at least $P + 1$ compartments, or variables, denoted $(S_p)_{p \leq P}$ and R :

$$\begin{cases} S'_p(t) &= F_{S_p}(S_p(t), R(t), \boldsymbol{\psi}), \quad p \leq P, \\ R'(t) &= F_R(S_p(t), R(t), \boldsymbol{\psi}), \end{cases} \quad (1)$$

where $\boldsymbol{\psi}$ is an m -dimensional parameter vector, and F_{S_p} , $p \leq P$, and F_R are fixed parametric functions characterizing the mechanistic process. For each individual i , $i = 1, \dots, N$, the trajectories generated by this ODE system are written S_{pi} ($p \leq P$) and R_i , with individual-specific parameters $\boldsymbol{\psi}_i = (\psi_{li})_{l \leq m}$. Each parameter ψ_{li} ($l = 1, \dots, m$) is modeled through a generalized linear model:

$$h_l(\psi_{li}) = h_l(\psi_{lpop}) + \mathbf{X}_i \boldsymbol{\beta}_l + \eta_{li},$$

where \mathbf{X}_i denotes the vector of individual covariates with regression coefficients β_l , and the random effects satisfy $\boldsymbol{\eta}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega})$. In particular, it is possible to fix some parameters to a common value across individuals by setting the corresponding variances in $\boldsymbol{\Omega}$ to zero.

The individual compartments S_{pi} , $p \leq P$, are observed through repeated measurements Y_{pij} at time points t_{pij} , with $j \leq n_{pi}$:

$$Y_{pij} = g_p(S_{pi}(t_{pij})) + \epsilon_{pij}, \quad (2)$$

for a fixed transformation g_p , with independent errors $\epsilon_{pij} \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \varsigma_p^2)$.

The latent compartment R_i is measured repeatedly through a large number of markers Z_{kij} for $k \leq K$, at times t_{kij} , $j \leq n_{ki}$, via a linear observation model:

$$Z_{kij} = \mu_k + \alpha_k s_k(R_i(t_{kij})) + \varepsilon_{kij}, \quad (3)$$

with fixed transformation s_k and independent errors $\varepsilon_{kij} \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma_k^2)$. We further denote the observation vectors $\mathbf{Y}_{pi} = (Y_{pij})_{j \leq n_{pi}}$ for $p = 1, \dots, P$ and $\mathbf{Z}_{ki} = (Z_{kij})_{j \leq n_{ki}}$ for $k = 1, \dots, K$.

with a fixed transformation s_k and independent measurement errors $\varepsilon_{kij} \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma_k^2)$. We denoted later $\mathbf{Y}_{pi} = (Y_{pij})_{j \leq n_{pi}}$, $p = 1, \dots, P$, and $\mathbf{Z}_{ki} = (Z_{kij})_{j \leq n_{ki}}$, $k = 1, \dots, K$, the compartment observation vectors for individual i .

For the remainder of the manuscript, we distinguish between the large-dimensional biomarker slopes $\boldsymbol{\alpha} = (\alpha_k)_{k \leq K}$, the coefficients linking markers to the latent process and subject to regularization, and the vector $\boldsymbol{\theta} = (\boldsymbol{\psi}_{\text{pop}}, \boldsymbol{\beta}, \boldsymbol{\Omega}, (\sigma_k^2)_{k \leq K}, (\varsigma_p^2)_{p \leq P}, (\mu_k)_{k \leq K})$, which gathers all remaining model parameters.

2.1.2 Log-likelihood of the model

We denote the individual observations for subject $i = 1, \dots, N$ as $\mathbf{Y}_i = (\mathbf{Y}_{pi})_{p \leq P}$ from Equation (2) and $\mathbf{Z}_i = (\mathbf{Z}_{ki})_{k \leq K}$ from Equation (3). For individual i , the contribution to the likelihood is:

$$\mathcal{L}_i(\boldsymbol{\alpha}, \boldsymbol{\theta}) = f(\mathbf{Y}_i, \mathbf{Z}_i \mid \boldsymbol{\alpha}, \boldsymbol{\theta}) = \int_{\boldsymbol{\eta}_i} f(\mathbf{Y}_i, \mathbf{Z}_i \mid \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) \times f(\boldsymbol{\eta}_i \mid \boldsymbol{\alpha}, \boldsymbol{\theta}) d\boldsymbol{\eta}_i.$$

Conditionally on the individual random effects, the trajectories $(S_{pi})_{p \leq P}$ and R_i are determined by the differential system (1). Hence, Z_{kij} ($k = 1, \dots, K$; $j = 1, \dots, n_{ki}$) are independent with distribution $\mathcal{N}(\mu_k + \alpha_k s_k(R_i(t_{kij})), \sigma_k^2)$, and Y_{pij} ($p = 1, \dots, P$; $j = 1, \dots, n_{pi}$) are independent with distribution $\mathcal{N}(g_p(S_{pi}(t_{pij})), \varsigma_p^2)$. We then have

$$\begin{aligned} \mathcal{L}_i(\boldsymbol{\alpha}, \boldsymbol{\theta}) &= \int_{\boldsymbol{\eta}_i} \left\{ \prod_{p \leq P} f(\mathbf{Y}_{pi} \mid \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) \right\} \times \left\{ \prod_{k \leq K} f(\mathbf{Z}_{ki} \mid \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) \right\} \times f(\boldsymbol{\eta}_i \mid \boldsymbol{\alpha}, \boldsymbol{\theta}) d\boldsymbol{\eta}_i \\ &= \int_{\boldsymbol{\eta}_i} \left\{ \prod_{p \leq P} \prod_{j \leq n_{pi}} f(Y_{pij} \mid \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) \right\} \times \left\{ \prod_{k \leq K} \prod_{j \leq n_{ki}} f(Z_{kij} \mid \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) \right\} \times f(\boldsymbol{\eta}_i \mid \boldsymbol{\alpha}, \boldsymbol{\theta}) d\boldsymbol{\eta}_i \end{aligned}$$

$$\begin{aligned}
&= \int_{\boldsymbol{\eta}_i} \left\{ \prod_{p \leq P} \prod_{j \leq n_{pi}} \frac{1}{\varsigma_p \sqrt{2\pi}} \exp \left[-\frac{1}{2} \left(\frac{Y_{pij} - g_p(S_{pi}(t_{pij}))}{\varsigma_p} \right)^2 \right] \right\} \\
&\times \left\{ \prod_{k \leq K} \prod_{j \leq n_{ki}} \frac{1}{\sigma_k \sqrt{2\pi}} \exp \left[-\frac{1}{2} \left(\frac{Z_{kij} - \mu_k - \alpha_k s_k(R_i(t_{kij}))}{\sigma_k} \right)^2 \right] \right\} \\
&\times \left\{ \frac{1}{(2\pi)^{\frac{m}{2}} \det \boldsymbol{\Omega}^{\frac{1}{2}}} \exp \left[-\frac{1}{2} \boldsymbol{\eta}_i^T \boldsymbol{\Omega}^{-1} \boldsymbol{\eta}_i \right] \right\} d\boldsymbol{\eta}_i.
\end{aligned}$$

Thus, the individual likelihood can be expressed as a product of: (1) the density of \mathbf{Z}_i , $f_Z(\mathbf{Z}_i \mid \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i)$; (2) the density of \mathbf{Y}_i , $f_Y(\mathbf{Y}_i \mid \boldsymbol{\theta}, \boldsymbol{\eta}_i)$ (which does not depend on $\boldsymbol{\alpha}$); and (3) the density of the random effects $f(\boldsymbol{\eta}_i \mid \boldsymbol{\theta})$. With independent individuals, the log-likelihood is:

$$LL(\boldsymbol{\alpha}, \boldsymbol{\theta}) = \sum_{i \leq N} \log \left(\int_{\boldsymbol{\eta}_i} f_Y(\mathbf{Y}_i \mid \boldsymbol{\theta}, \boldsymbol{\eta}_i) f_Z(\mathbf{Z}_i \mid \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i \mid \boldsymbol{\theta}) d\boldsymbol{\eta}_i \right). \quad (4)$$

The calculation of the log-likelihood and its derivatives is detailed in Section 2.3.2.

2.2 Overview of the estimation process

REMixed aims to estimate both the structural parameters $\boldsymbol{\theta}$ and the sparse vector of biomarker effects $\boldsymbol{\alpha}$ by maximizing the penalized log-likelihood under Lasso-penalization:

$$LL_{\text{pen}_\lambda}(\boldsymbol{\alpha}, \boldsymbol{\theta}) = LL(\boldsymbol{\alpha}, \boldsymbol{\theta}) - \lambda \|\boldsymbol{\alpha}\|_1, \quad (5)$$

for a given penalty $\lambda > 0$, selected by minimizing BICc (Delattre et al., 2014) over a grid (see Section 2.3.4). Estimation combines penalized maximum likelihood with a cyclical coordinate descent scheme. Each iteration alternates between:

1. **Regularization step:** with $\boldsymbol{\theta}$ fixed at the previous iteration estimates, update $\boldsymbol{\alpha}$ by solving a Lasso-type problem based on a second-order Taylor approximation of the log-likelihood around the current $\boldsymbol{\alpha}$. This yields a closed-form update for each α_k involving the gradient $\partial_{\boldsymbol{\alpha}} LL$, the Hessian $\partial_{\boldsymbol{\alpha}}^2 LL$, and the penalty λ (see Section 2.3.1).
2. **Estimation step:** with $\boldsymbol{\alpha}$ fixed at its updated value, estimate $\boldsymbol{\theta}$ using the SAEM algorithm (Delyon et al., 1999; Kuhn and Lavielle, 2005), implemented in Monolix and accessed from R via **lioftConnectors** (see Section 2.3.3).

Convergence is reached when the penalized log-likelihood and the parameters stabilize below user-defined tolerances τ_1 and τ_2 , respectively. The pipeline is shown in Figure 1, and pseudocode is given in Algorithm 1.

Algorithm 1: REMixed algorithm:

Let $\lambda > 0$ be fixed ;

Obtain initial values $\alpha^{(0)}, \theta^{(0)}$, e.g., by MLE or SAEM on a random subset of biomarkers (see Section 2.3.6).

Estimate the posterior distribution of random effects via MCMC in Monolix.

Compute $LL(\alpha^{(0)}, \theta^{(0)})$, $\partial_\alpha LL(\alpha, \theta^{(0)})|_{\alpha=\alpha^{(0)}}$, and $\partial_\alpha^2 LL(\alpha, \theta^{(0)})|_{\alpha=\alpha^{(0)}}$.

At iteration l :

Compute $\alpha^{(l+1)}$ using Equation (6) with $\hat{\alpha} = \alpha^{(l)}$ and $\hat{\theta} = \theta^{(l)}$ fixed (Section 2.3.1).

Estimate $\theta^{(l+1)}$ with α fixed at $\alpha^{(l+1)}$ using SAEM initialized at $\theta^{(l)}$.

Update $LL(\alpha^{(l+1)}, \theta^{(l+1)})$, $\partial_\alpha LL(\alpha, \theta^{(l+1)})|_{\alpha=\alpha^{(l+1)}}$, and

$\partial_\alpha^2 LL(\alpha, \theta^{(l+1)})|_{\alpha=\alpha^{(l+1)}}$ using a new posterior approximation of the random effects.

Evaluate convergence and stop if all criteria are met:

$$\left| LL_{\text{pen}_\lambda}(\alpha^{(l+1)}, \theta^{(l+1)}) - LL_{\text{pen}_\lambda}(\alpha^{(l)}, \theta^{(l)}) \right| < \tau_1;$$

$$\left| \left(\frac{\alpha^{(l)} - \alpha^{(l+1)}}{\alpha^{(l)} + 1}, \frac{\theta^{(l)} - \theta^{(l+1)}}{\theta^{(l)} + 1} \right) \right| < \tau_2.$$

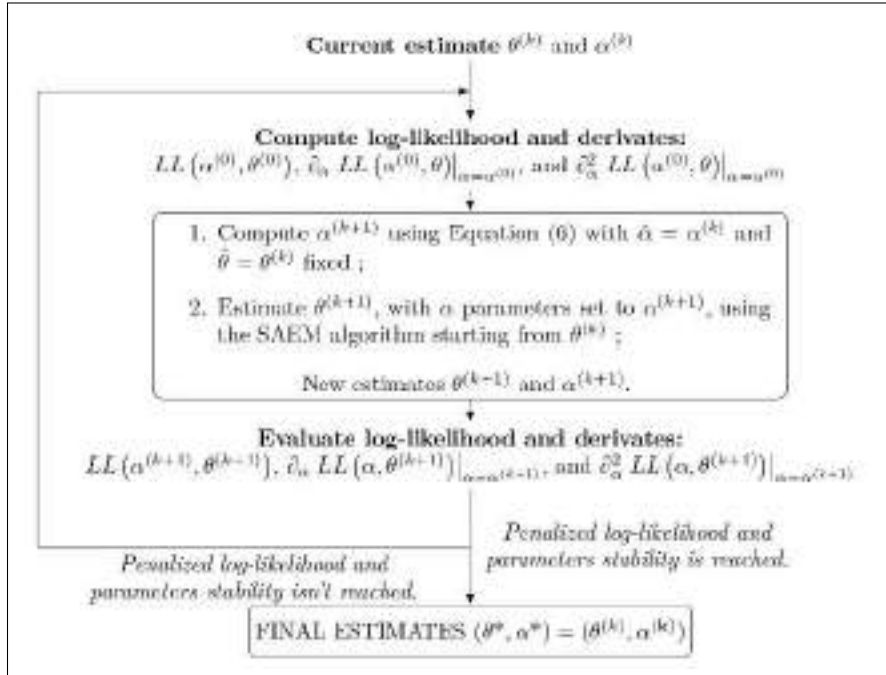


Figure 1: Pipeline of the REMixed algorithm.

2.3 Implementation details

2.3.1 Regularization step

Given $\hat{\boldsymbol{\theta}}$ and the current $\hat{\boldsymbol{\alpha}}$, maximizing (5) corresponds to solving the Lasso-penalized problem

$$\max_{\boldsymbol{\alpha}} \left\{ LL(\hat{\boldsymbol{\theta}}, \boldsymbol{\alpha}) - \lambda \|\boldsymbol{\alpha}\|_1 \right\}.$$

A second-order Taylor approximation of $LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})$ around $\hat{\boldsymbol{\alpha}}$ yields

$$LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}}) \approx L(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\theta}}) + (\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}})^T \frac{\partial}{\partial \boldsymbol{\alpha}} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}}) \Big|_{\boldsymbol{\alpha}=\hat{\boldsymbol{\alpha}}} + \frac{1}{2} (\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}})^T \frac{\partial^2}{\partial \boldsymbol{\alpha}^2} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}}) \Big|_{\boldsymbol{\alpha}=\hat{\boldsymbol{\alpha}}} (\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}).$$

Let's define for $k = 1, \dots, K$, $A_k = \partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}}) \Big|_{\hat{\boldsymbol{\alpha}}} + \hat{x}_{kk} \hat{\alpha}_k$, where \hat{x}_{kk} is the k th diagonal element of the opposite of the Hessian at $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\theta}})$, assumed negative definite. The updates follow the standard soft-thresholding rule:

$$\alpha_k = \begin{cases} \frac{A_k + \lambda}{\hat{x}_{kk}} & \text{if } A_k < -\lambda, \\ \frac{A_k - \lambda}{\hat{x}_{kk}} & \text{if } A_k > \lambda, \\ 0 & \text{otherwise.} \end{cases} \quad \text{with } A_k = \partial_{\alpha_k} LL(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\alpha}}) + \hat{x}_{kk} \hat{\alpha}_k \quad (6)$$

Additional derivations and proofs are provided in Appendix A and log-likelihood and derivative evaluations are detailed in the next Section 2.3.2.

To ensure numerical stability in case of model misspecification, the Hessian may fail to be negative definite; in that case we inflate its diagonal by adding a scaled identity matrix, in the spirit of the Marquardt–Levenberg method (Philipps et al., 2021). The pseudocode of the regularization step appears in Algorithm 2.

Algorithm 2: Regularization step at iteration l , with penalty parameter $\lambda > 0$:

Let $\boldsymbol{\theta}^{(l)}$ be fixed, and $\boldsymbol{\alpha}^{(l)}$ be the current estimate ;

Compute $LL(\boldsymbol{\alpha}^{(l)}, \boldsymbol{\theta}^{(l)})$, $\partial_{\boldsymbol{\alpha}} LL(\boldsymbol{\alpha}, \boldsymbol{\theta}^{(l)}) \Big|_{\boldsymbol{\alpha}=\boldsymbol{\alpha}^{(l)}}$, and $\partial_{\boldsymbol{\alpha}}^2 LL(\boldsymbol{\alpha}, \boldsymbol{\theta}^{(l)}) \Big|_{\boldsymbol{\alpha}=\boldsymbol{\alpha}^{(l)}} = -(x_{lc})_{l,c \leq K}$ and inflate the Hessian if needed.

For $k \leq K$:

 Compute $A_k = \partial_{\alpha_k} LL(\boldsymbol{\alpha}^{(l)}, \boldsymbol{\theta}^{(l)}) + x_{kk} \alpha_k^{(l)}$.

if $A_k < -\lambda$:

if $A_k < -\lambda$: $\alpha_k \leftarrow (A_k + \lambda)/x_{kk}$.

else if $A_k > \lambda$: $\alpha_k \leftarrow (A_k - \lambda)/x_{kk}$.

else: $\alpha_k \leftarrow 0$.

Compute $LL(\boldsymbol{\alpha}, \boldsymbol{\theta}^{(l)})$ with the updated $\boldsymbol{\alpha}$.

If $LL_{\text{pen}_{\lambda}}(\boldsymbol{\alpha}^{(l)}, \boldsymbol{\theta}^{(l)}) > LL_{\text{pen}_{\lambda}}(\boldsymbol{\alpha}, \boldsymbol{\theta}^{(l)})$,

recalibrate $\boldsymbol{\alpha}$ by searching for a better candidate in its neighborhood.

2.3.2 Log-likelihood estimation

2.3.2.1 Log-likelihood derivatives

The log-likelihood of the model is given in Section 2.1.2. We now derive the first and second derivatives with respect to the coordinate α used in REMixed (see Figure 1 and Section 2.3.1). For every $k \leq K$,

$$\frac{\partial f_Y(\mathbf{Y}_i | \boldsymbol{\theta}, \boldsymbol{\eta}_i) f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \boldsymbol{\theta})}{\partial \alpha_k} = \frac{\partial f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i)}{\partial \alpha_k} \times f_Y(\mathbf{Y}_i | \boldsymbol{\theta}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \boldsymbol{\theta}),$$

and

$$\frac{\partial f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i)}{\partial \alpha_k} = \frac{1}{\sigma_k^2} \left\{ \sum_{j \leq n_{ki}} s_k(R_i(t_{kij})) (Z_{kij} - \mu_k - \alpha_k s_k(R_i(t_{kij}))) \right\} \times f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i).$$

Similarly,

$$\frac{\partial^2 f_Y(\mathbf{Y}_i | \boldsymbol{\theta}, \boldsymbol{\eta}_i) f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \boldsymbol{\theta})}{\partial \alpha_k^2} = \frac{\partial^2 f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i)}{\partial \alpha_k^2} \times f_Y(\mathbf{Y}_i | \boldsymbol{\theta}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \boldsymbol{\theta}),$$

and

$$\begin{aligned} & \frac{\partial^2 f_Y(\mathbf{Y}_i | \boldsymbol{\theta}, \boldsymbol{\eta}_i) f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \boldsymbol{\theta})}{\partial \alpha_k^2} \\ &= \left(\frac{1}{\sigma_k^4} \left(\sum_{j \leq n_{ki}} s_k(R_i(t_{kij})) (Z_{kij} - \mu_k - \alpha_k s_k(R_i(t_{kij}))) \right)^2 - \frac{1}{\sigma_k^2} \left(\sum_{j \leq n_{ki}} s_k(R_i(t_{kij}))^2 \right) \right) f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i). \end{aligned}$$

Finally, for every $k_1, k_2 \leq K$, $k_1 \neq k_2$:

$$\frac{\partial^2 f_Y(\mathbf{Y}_i | \boldsymbol{\theta}, \boldsymbol{\eta}_i) f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \boldsymbol{\theta})}{\partial \alpha_{k_1} \alpha_{k_2}} = \frac{\partial^2 f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i)}{\partial \alpha_{k_1} \alpha_{k_2}} \times f_Y(\mathbf{Y}_i | \boldsymbol{\theta}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \boldsymbol{\theta}),$$

and

$$\begin{aligned} & \frac{\partial^2 f_Y(\mathbf{Y}_i | \boldsymbol{\theta}, \boldsymbol{\eta}_i) f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \boldsymbol{\theta})}{\partial \alpha_{k_1} \alpha_{k_2}} \\ &= \prod_{k=k_1, k_2} \left\{ \frac{1}{\sigma_k^2} \left[\sum_{j \leq n_{ki}} s_k(R_i(t_{kij})) (Z_{kij} - \mu_k - \alpha_k s_k(R_i(t_{kij}))) \right] \right\} \times f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i). \end{aligned}$$

Therefore, for $k \leq K$, we can then derivate the log-likelihood with respect to α_k :

$$\begin{aligned} \frac{\partial LL(\boldsymbol{\alpha}, \boldsymbol{\theta})}{\partial \alpha_k} &= \sum_{i \leq N} \frac{\int_{\boldsymbol{\eta}_i} \partial_{\alpha_k} f_Y(\mathbf{Y}_i | \boldsymbol{\theta}, \boldsymbol{\eta}_i) f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \boldsymbol{\theta}) d\boldsymbol{\eta}_i}{\mathcal{L}_i(\boldsymbol{\alpha}, \boldsymbol{\theta})} \\ &= \sum_{i \leq N} \frac{1}{\mathcal{L}_i(\boldsymbol{\alpha}, \boldsymbol{\theta})} \times \frac{\partial f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i)}{\partial \alpha_k} \times f_Y(\mathbf{Y}_i | \boldsymbol{\theta}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \boldsymbol{\theta}), \end{aligned} \quad (7)$$

and

$$\begin{aligned} \frac{\partial^2 LL(\boldsymbol{\alpha}, \boldsymbol{\theta})}{\partial \alpha_k^2} &= - \left(\frac{\partial LL_i(\boldsymbol{\alpha}, \boldsymbol{\theta})}{\partial \alpha_k} \right)^2 \\ &+ \sum_{i \leq N} \frac{1}{\mathcal{L}_i(\boldsymbol{\alpha}, \boldsymbol{\theta})} \times \partial_{\alpha_k}^2 f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) \times f_Y(\mathbf{Y}_i | \boldsymbol{\theta}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \boldsymbol{\theta}) d\boldsymbol{\eta}_i, \end{aligned} \quad (8)$$

and, for $k_1 \neq k_2$,

$$\begin{aligned} \frac{\partial^2 LL(\boldsymbol{\alpha}, \boldsymbol{\theta})}{\partial \alpha_{k_1} \partial \alpha_{k_2}} &= - \frac{\partial LL_i(\boldsymbol{\alpha}, \boldsymbol{\theta})}{\partial \alpha_{k_1}} \times \frac{\partial LL_i(\boldsymbol{\alpha}, \boldsymbol{\theta})}{\partial \alpha_{k_2}} \\ &+ \sum_{i \leq N} \frac{1}{\mathcal{L}_i(\boldsymbol{\alpha}, \boldsymbol{\theta})} \times \partial_{\alpha_{k_1} \alpha_{k_2}} f_Z(\mathbf{Z}_i | \boldsymbol{\alpha}, \boldsymbol{\theta}, \boldsymbol{\eta}_i) \times f_Y(\mathbf{Y}_i | \boldsymbol{\theta}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \boldsymbol{\theta}) d\boldsymbol{\eta}_i. \end{aligned} \quad (9)$$

2.3.2.2 Log-likelihood approximation by Gauss–Hermite quadrature

The integrals in Equations (4), (7), (8), and (9) are evaluated at $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\theta}})$ using adaptive Gauss–Hermite quadrature, combined with numerical ODE solvers to compute R_i and $(S_{pi})_p$ for $i = 1, \dots, N$ jointly across all integrals. We first use Monolix to approximate the posterior distribution of the random effects, which concentrates the mass of the integrands.

The posterior $f(\boldsymbol{\eta}_i | \mathbf{Y}_i, \mathbf{Z}_i, \boldsymbol{\alpha}, \boldsymbol{\theta})$ of the individual random effects $\boldsymbol{\eta}_i$ given the data, is approximated in Monolix by a Metropolis–Hastings algorithm (Hastings, 1970). For each subject i , a chain $\{\boldsymbol{\eta}_i^{(m)}\}_{m=1}^M$ targeting this posterior is generated using a Gaussian proposal; a candidate $\boldsymbol{\eta}_i^*$ is accepted with a probability depending on the proposal density, the observation density $f(\mathbf{Y}_i, \mathbf{Z}_i | \boldsymbol{\eta}_i, \boldsymbol{\alpha}, \boldsymbol{\theta})$, and the prior $f(\boldsymbol{\eta}_i | \boldsymbol{\theta})$. After a burn-in phase to reach stationarity, the empirical mean and covariance of a Gaussian approximation to the posterior are computed from the last M accepted samples $(\boldsymbol{\eta}_i^{(m)})$.

We then compute multidimensional adaptive Gauss–Hermite quadrature to evaluate the integrals in (4), (7), (8), and (9) at $(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\alpha}})$.

Log-likelihood and derivative evaluations are carried out at each iteration by the function ‘gh.LL’ in the **REMixed** package. The pseudocode of this step is given in Algorithm 3.

Algorithm 3: Log-likelihood and derivates computation:

Let $\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\theta}}$ be fixed ;

Approximate the Gaussian posterior $f(\boldsymbol{\eta}_i | \mathbf{Y}_i, \mathbf{Z}_i, \boldsymbol{\alpha}, \boldsymbol{\theta})$ via MCMC in Monolix.

For each individual $i = 1, \dots, N$ (can be execute in parallel):

 Compute the adaptive Gauss–Hermite grid $(\boldsymbol{\eta}_{il}, w_l)_{l \leq m}$ from the Gaussian posterior approximation.

 Evaluate the integrands of the log-likelihood and derivatives at $\boldsymbol{\eta}_{il}$ for $l \leq m$, with m the number of quadrature points, and take the corresponding weighted sums.

 For instance, for the individual likelihood, are computed:

$$f_Z(\mathbf{Z}_i | \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\theta}}, \boldsymbol{\eta}_{il}), f_Y(\mathbf{Y}_i | \hat{\boldsymbol{\theta}}, \boldsymbol{\eta}_{il}), f(\boldsymbol{\eta}_{il} | \hat{\boldsymbol{\theta}})$$

 and then $\sum_{l \leq m} w_l f_Z(\mathbf{Z}_i | \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\theta}}, \boldsymbol{\eta}_{il}) f_Y(\mathbf{Y}_i | \hat{\boldsymbol{\theta}}, \boldsymbol{\eta}_{il}) f(\boldsymbol{\eta}_{il} | \hat{\boldsymbol{\theta}})$ is returned as $\hat{\mathcal{L}}_i(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\theta}})$.

Aggregate individual contributions to obtain model log-likelihood and derivatives.

2.3.3 NLME parameters estimation

Population parameters $\boldsymbol{\theta}$ are estimated at each iteration via the SAEM algorithm (Delyon et al., 1999; Kuhn and Lavielle, 2005), as implemented in Monolix. During this step, $\boldsymbol{\alpha}$ is held fixed at the previously estimated value $\hat{\boldsymbol{\alpha}}$ (see Figure 1).

The SAEM is an iterative algorithm for maximum-likelihood estimation in models with latent variables, particularly well-suited to NLMEMs. Each iteration l alternates:

- **a Stochastic Simulation step (S-step):** sample the individual random effects $\boldsymbol{\eta}^{(l)}$ from their conditional distribution given the data and current parameter estimates, i.e., $\boldsymbol{\eta}^{(l)} \sim f(\boldsymbol{\eta} \mid \mathbf{Y}, \mathbf{Z}, \boldsymbol{\theta}^{(l)})$, to approximate the expected complete-data log-likelihood $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(l)})$.
- **a Maximization step (M-step):** update $\boldsymbol{\theta}$ by maximizing the approximation of $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(l)})$ obtained in the S-step.

The SAEM algorithm has been demonstrated to be highly efficient and reliable for a broad scope of complex models, including those involving categorical data, count data, time-to-event data, mixture models, and models based on differential equations or censored data. Its convergence properties have been rigorously established, ensuring robustness in parameter estimation.

In our implementation, Monolix is interfaced with R via **lixoftConnectors**. In **REMixed**, the SAEM and Lasso updates alternate until convergence in the ‘**remix**’ function for a fixed penalty, or across a grid via ‘**cv.remix**’.

2.3.4 Selection of the penalty parameter

To select the optimal penalty λ , we run the algorithm over the grid

$$\Lambda = \left\{ \lambda_{\max} \times \beta_{\lambda}^{l/N_{\lambda}}, l = 1, \dots, N_{\lambda} \right\},$$

where N_{λ} is the grid size and β_{λ} is user-defined (typically 0.001 (Park and Hastie, 2007; Friedman et al., 2010)), and

$$\lambda_{\max} = \max \left(\partial_{\boldsymbol{\alpha}} LL(\boldsymbol{\alpha}, \boldsymbol{\theta}^{(0)})|_{\boldsymbol{\alpha}=\mathbf{0}} \right). \quad (10)$$

The final model is chosen by minimizing the corrected Bayesian Information Criterion (BICc, Delattre et al. (2014)), accounting for fixed and random effects and for the number of selected biomarkers:

$$\text{BICc}(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\theta}}) = -2LL(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\theta}}) + \log(N) \dim(\boldsymbol{\theta}_R) + \log(n_{\text{tot}}) (\dim(\boldsymbol{\theta}_F) + \#\{\alpha_k \neq 0\}),$$

where $\boldsymbol{\theta}$ is decomposed as $(\boldsymbol{\theta}_R, \boldsymbol{\theta}_F)$, the components related to random and fixed effects, respectively.

Let $\hat{\boldsymbol{\theta}}_{\lambda}$ and $\hat{\boldsymbol{\alpha}}_{\lambda}$ denote the REMixed estimates for a given $\lambda \in \Lambda$. We then select

$$\lambda^* = \arg \min_{\lambda \in \Lambda} \text{BICc}(\hat{\boldsymbol{\alpha}}_{\lambda}, \hat{\boldsymbol{\theta}}_{\lambda}), \quad (\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\theta}}) = (\hat{\boldsymbol{\alpha}}_{\lambda^*}, \hat{\boldsymbol{\theta}}_{\lambda^*}).$$

To compute λ_{\max} defined in (10), the gradient in Section 2.3.2 simplifies. For $k \leq K$ and $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$,

$$f_Z(\mathbf{Z}_i | \boldsymbol{\alpha} = \mathbf{0}, \hat{\boldsymbol{\theta}}, \boldsymbol{\eta}_i) = \prod_{k \leq K} \prod_{j \leq n_{ki}} \frac{1}{\sigma_k \sqrt{2\pi}} \exp \left(-\frac{1}{2} \left(\frac{Z_{kij} - \mu_k}{\sigma_k} \right)^2 \right),$$

does not depend on $\boldsymbol{\eta}_i$. Hence, for $i = 1, \dots, N$,

$$\mathcal{L}_i(\hat{\boldsymbol{\theta}}, \mathbf{0}) = f_Z(\mathbf{Z}_i | \hat{\boldsymbol{\theta}}, \boldsymbol{\alpha} = \mathbf{0}) \int_{\boldsymbol{\eta}_i} f_Y(\mathbf{Y}_i | \hat{\boldsymbol{\theta}}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \hat{\boldsymbol{\theta}}) d\boldsymbol{\eta}_i,$$

and therefore $\partial_{\alpha_k} LL(\hat{\boldsymbol{\theta}}, \boldsymbol{\alpha})|_{\boldsymbol{\alpha}=\mathbf{0}}$ is equal to

$$\sum_{i \leq N} \frac{1}{\int_{\boldsymbol{\eta}_i} f_Y(\mathbf{Y}_i | \hat{\boldsymbol{\theta}}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \hat{\boldsymbol{\theta}}) d\boldsymbol{\eta}_i} \times \int_{\boldsymbol{\eta}_i} \frac{1}{\sigma_k^2} \left(\sum_{j \leq n_{ki}} s_k(R_i(t_{kij})) (Z_{kij} - \mu_k) \right) \times f_Y(\mathbf{Y}_i | \hat{\boldsymbol{\theta}}, \boldsymbol{\eta}_i) f(\boldsymbol{\eta}_i | \hat{\boldsymbol{\theta}}) d\boldsymbol{\eta}_i.$$

2.3.5 Final tests

After convergence, we assess the statistical significance of each estimated coefficient $\hat{\alpha}_k$ using Wald tests based on the asymptotic normality of maximum-likelihood estimators, testing $H_0 : \alpha_k = 0$.

This procedure allows us to identify the parameters that contribute meaningfully to the model, providing insight into relevant biomarkers.

These tests can be launch in **REMixed** using ‘computeFinalTest’ function.

2.3.6 Initialization strategy

To reduce computation time, we propose an initialization that sets $\boldsymbol{\theta}^{(0)}$ and $\boldsymbol{\alpha}^{(0)}$ from randomly partitioning the K biomarkers into Γ groups $(\mathcal{G}_\gamma)_{\gamma \leq \Gamma}$ of approximately equal size, fitting each group-specific model independently, and selecting the best.

Specifically, for each group \mathcal{G}_γ , define the model \mathcal{M}_γ as follows:

- For $k \notin \mathcal{G}_\gamma$, the marker is considered as Gaussian noise (i.e., not selected). We set $\alpha_k = 0$ and use the simplified observation model

$$Z_{kij} = \mu_k + \varepsilon_{kij}, \quad \varepsilon_{kij} \sim \mathcal{N}(0, \sigma_k^2).$$

The paramerers (μ_k, σ_k^2) are then estimated via Maximum Likelihood Estimation (MLE) using

$$\hat{\mu}_k = \overline{Z_{kij}}^{i,j}, \quad \hat{\sigma}_k^2 = \frac{1}{\sum_{i=1}^N n_i} \sum_{i=1}^N \sum_{j=1}^{n_i} \left(Z_{kij} - \overline{Z_{kij}}^{i,j} \right)^2, \quad (11)$$

where $\overline{Z_{kij}}^{i,j}$ denotes the overall mean of Z_{kij} across subjects and times.

- For $k \in \mathcal{G}_\gamma$, we use the full observation equation and estimate α_k jointly with $\boldsymbol{\theta}$ via the SAEM algorithm.

Finally, the total log-likelihood LL_γ of the fitted model \mathcal{M}_γ , considering the K markers of \mathcal{G}_γ and $\overline{\mathcal{G}}_\gamma$, is then computed via importance sampling using Monolix software. At this stage, gradient and Hessian evaluations are not required, making this estimation computationally possible through Monolix, contrary to later estimation of the log-likelihood (see details in Section 2.3.2). The group \mathcal{G} that yields the highest log-likelihood is then selected, and the corresponding estimates are used to initialize the REMixed algorithm.

This initialization is available through ‘`initStrat`’ function in the **REMixed** package, and its pseudocode is provided in Algorithm 4. Its used is illustrated in Section 4.

Algorithm 4: Initialization strategy:

Randomly split biomarkers into Γ groups \mathcal{G}_γ of (approximately) equal size.

For each $\gamma \leq \Gamma$:

 Compute MLEs $(\hat{\mu}_k, \hat{\sigma}_k)_{k \notin \mathcal{G}_\gamma}$ as in (11).

 With $(\hat{\mu}_k, \hat{\sigma}_k, \hat{\alpha}_k = 0)_{k \notin \mathcal{G}_\gamma}$ fixed, estimate θ and $(\alpha_k)_{k \in \mathcal{G}_\gamma}$ by SAEM.

 Compute the log-likelihood LL_γ .

Set $(\alpha^{(0)}, \theta^{(0)})$ to the estimates from \mathcal{M}_G with $G = \arg \max_{\gamma \leq \Gamma} LL_\gamma$.

3 Project architecture and connection with Monolix

3.1 Monolix project

The **REMixed** package relies on the Monolix software suite for nonlinear mixed-effects model estimation, interfaced through the **lixoftConnectors** R package [R Core Team \(2024\)](#). This package provides a direct link between R and the MonolixSuite modeling tools, enabling users to automate workflows, control MonolixSuite software, and perform model development and simulation processes.

To initialize the Monolix connection in R, the user needs to load the library and call the initialization function:

```
library(lixoftConnectors)
initializeLixoftConnectors(software = "monolix")
```

Once initialized, the **lixoftConnectors** functions allow creating, editing, and running Monolix projects from R. A Monolix project is defined by a set of key files:

- ‘data.txt’ – the dataset used for estimation,
- ‘model.txt’ – the structural model, written in Mlxtran syntax,
- ‘project.mlxtran’ – the main project file specifying model settings,
- ‘project.mlxproperties’ – optional preferences for visualization.

See [Lixoft SAS, a Simulations Plus company \(2023a\)](#) for details about the Monolix software.

When a task is executed (e.g., running the SAEM algorithm or computing conditional distributions), Monolix creates a results folder within the project directory, including various output files such as individual predictions and estimation summaries.

After executing a task, a subfolder is created with output files including ‘summary.txt’, which contains estimation and diagnostic results.

3.2 Summary files and temporary projects

During execution, the **REMixed** package creates temporary Monolix projects and stores summary files for traceability. These are saved in a dedicated ‘remix’ folder within the Monolix project directory.

- **Initialization step** (via ‘initStrat’): temporary Monolix projects are stored in the ‘tmp_init’ folder. These are deleted by default to save space but can be preserved by setting `unlinkTemporaryProject = FALSE`. A log file called ‘summary_init.txt’ is saved in the ‘remix’ folder, summarizing the full initialization run (see Appendix D).
- **REMixed algorithm for a grid of penalty parameters** (via ‘cv.remix’): one summary file per penalty value is saved as ‘summary_l.txt’ (where l is the index of the penalty λ_l), and a global summary file ‘summary_cv.txt’ is also created (see Appendix E). Temporary Monolix projects corresponding to each penalty value are named ‘Build_l.mlxtran’, and are deleted by default unless `unlinkBuildProject = FALSE`. The summary file produced for each penalty parameter has the same format as the one provided for a single penalty parameter through the ‘remix’ function.
- **REMixed algorithm for a given penalty parameter** (via ‘remix’): a summary file named ‘summary.txt’ is created to monitor the iterations (see Appendix F). A copy of the initial project is also saved in the ‘remix’ folder to allow consistent reruns. The final Monolix project is saved in the same directory as the original input project.

After computing final tests and running the SAEM algorithm, the final ‘summary.txt’ is saved in the ‘remix’ folder of the specified final project. For instance, in the ‘project_upd’ folder (see Appendix B).

4 Usage

4.1 Demo project overview

We illustrate the use of the **REMixed** package through a demo project in a low-dimensional setting for fast computation. To access the Monolix project provided with the package:

```
library(ggplot2)
theme_set(theme_bw())
```

```

library(REMixed)
library(lixoftConnectors)
initializeLixoftConnectors()

project <- getMLXdir()
loadProject(project)

```

This example is based on a simulated antibody production model describing the dynamics over the 7 days following vaccine injection, during which antibody-secreting cells (ASC, denoted S) reach their peak and subsequently decline (Pasin et al., 2019). These cells produce antibodies (AB) at a rate φ_S , while S and AB decay at rates δ_S and δ_{AB} , respectively. The simulation was generated using Simulx (Lixoft SAS, a Simulations Plus company, 2023b), and the corresponding Monolix project is provided in the directory returned by `getMLXdir()`. For the structural model, written in Mlxtran syntax, is provided in Appendix C (See Lixoft SAS, a Simulations Plus company (2023a) for details about the Monolix software and Mlxtran syntax).

The mechanistic model is defined for subject $i = 1, \dots, N$ as:

$$\left\{ \begin{array}{l} \frac{dS_i}{dt}(t) = -\delta_{S_i} S_i(t), \\ \frac{dAB_i}{dt}(t) = \varphi_{S_i} S_i(t) - \delta_{AB_i} AB_i(t), \\ (S_i(0), AB_i(0)) = (0, 1000). \end{array} \right. \quad (12)$$

The individual parameters are assumed to follow log-normal distributions:

$$\left\{ \begin{array}{l} \log(\varphi_{S_i}) = \log(\varphi_{S_{pop}}) + \eta_i^{\varphi_S}, \\ \log(\delta_{AB_i}) = \log(\delta_{AB_{pop}}) + \eta_i^{\delta_{AB}}, \\ \log(\delta_{S_i}) = \log(\delta_{S_{pop}}) + \eta_i^{\delta_S}. \end{array} \right. \quad (13)$$

To simulate observations, antibody levels are measured at six time points t_{ij} , $j = 1, \dots, 6$, post-vaccination (days 0, 7, 21, 123, 180, 300), following:

$$Y_{ij} = \log_{10}(AB_i(t_{ij})) + \varepsilon_{ij}, \quad \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_{Ab}^2). \quad (14)$$

In parallel, five biomarkers $(G_k)_{k \leq 5}$ are observed daily from day 0 to 21, at times t_{kij} , $j = 1, \dots, 22$, with only one biomarker truly associated with the ASC compartment S :

$$G_{kij} = \alpha_k S_i(t_{kij}) + \epsilon_{kij}, \quad \epsilon_{kij} \sim \mathcal{N}(0, \sigma_{G_k}^2). \quad (15)$$

Parameter values used in the demo are summarized in Table 1. The simulated data are shown in Figure 2.

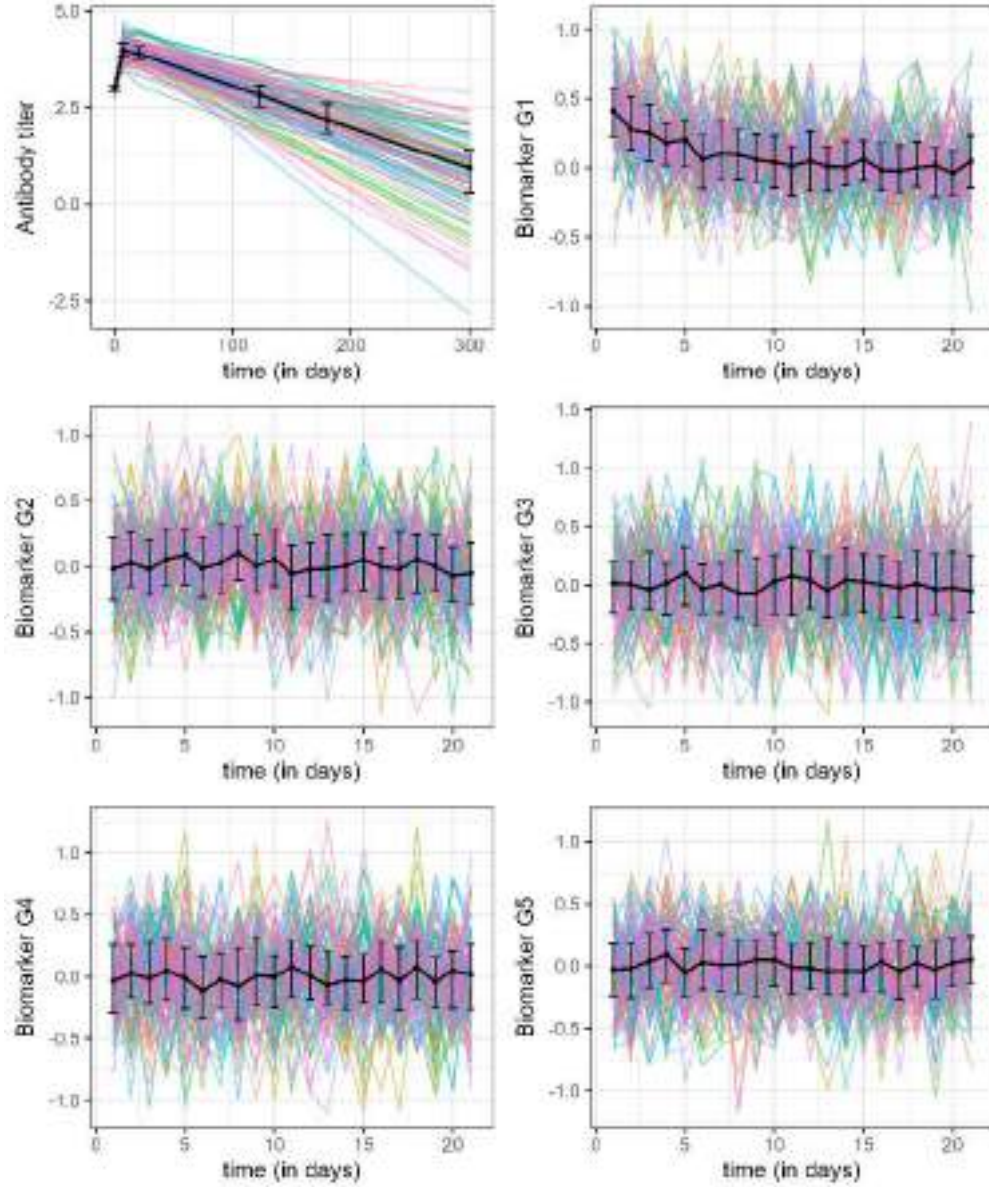


Figure 2: Simulated observations for 100 individuals from the demo project included in the **REMixed** package, representing antibody dynamics and biomarkers dynamics over 21 days following vaccination. In this scenario, only one of the five biomarkers, G_1 , is truly associated with the ASC compartment S .

Parameter	Definition	Value
Fixed Effects		
<i>Influx rates</i>		
φ_S	ASC-S antibody production (Eu/mL/day)	6.11×10^2
<i>Decay rates</i>		
δ_{Ab}	Antibody decay rate (1/day)	2.50×10^{-2}
δ_S	ASC-S decay rate	2.30×10^{-1}
Random Effects		
ω_{φ_S}	SD for φ_S	0.60
ω_{δ_S}	SD for δ_S	0.10
$\omega_{\delta_{Ab}}$	SD for δ_{Ab}	0.30
Biomarkers related parameters		
α_{11}		0.10
$\alpha_{1k}, k > 1$		0
Observation Error		
σ_{Ab}	SD of error on log antibody levels	0.11
σ_{G1}	SD of error on biomarkers levels	0.28
$\sigma_{Gk}, k > 1$		0.34, 0.39, 0.36, 0.32

Table 1: Parameter values used in the demo project simulation scenario, describing antibody and antibody-secreting cell (ASC) dynamics in 100 individuals. The scenario assumes five biomarkers $(G_k)_{k \leq 5}$, of which only G_1 is associated with the ASC compartment.

For **REMixed** to interface with Monolix, users must specify how to generate system dynamics and how to map outputs to observed data. This involves three elements:

1. The structural ODE model: it must return a table with time and named outputs. In the demo, ‘dynFUN_demo’ is pre-defined using the **deSolve** package (Karline Soetaert et al., 2010):

```
dynFUN_demo <- function(t, y, parms) {
  parms <- c(parms, S0 = y[["S"]])
  out <- deSolve::ode(y["AB"], t, function(t, y, parms) {
    with(as.list(c(y, parms)), {
      dAB <- phi_S * S0 * exp(-delta_S * t) - delta_AB * AB
      list(c(dAB))
    })
  }, parms)
  out <- cbind(out, S = y[["S"]] * exp(-parms[["delta_S"]] * out[, "time"]))
  class(out) <- "deSolve"
  return(out)
}

out <- dynFUN_demo(t = 1:300, y = c(S = 5, AB = 1000),
  parms = c(delta_S = 0.23, delta_AB = 0.025, phi_S = 611))
plot(out)
```

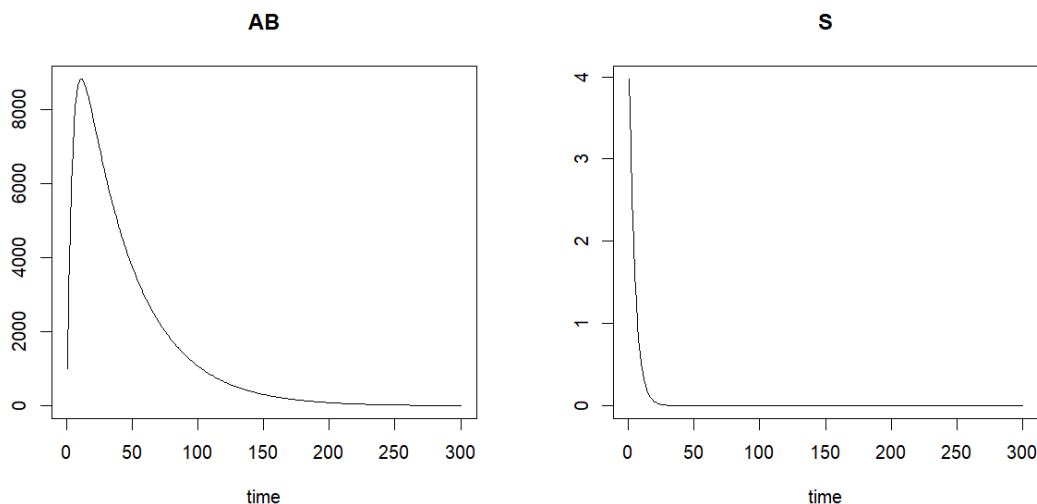


Figure 3: ”Deterministic dynamics of antibody and antibody-secreting cell (ASC) populations for the demo project, obtained by solving the system of differential equations with typical parameter values (random effects set to zero). These trajectories represent the expected population-level dynamics, without inter-individual variability.

2. The observation model: since Monolix does not store transformations used in the observation model, these must be explicitly specified via the ‘ObsModel.transfo’ object.

```

getContinuousObservationModel()$prediction
#   yAB   yG1   yG2   yG3   yG4   yG5
# "tAB"  "G1"  "G2"  "G3"  "G4"  "G5"

ObsModel.transfo = list(S = list(AB = log10),
                        linkS = "yAB",
                        R = rep(list(S = function(x){x}), 5),
                        linkR = paste0("yG", 1:5))

```

3. The biomarker Z_k statistical model: parameters must be explicitly linked to Monolix observation names. If no intercepts μ_k are included in the latent regression model, set `alpha0 = NULL`. Here ‘alpha0’ corresponds to the $\boldsymbol{\mu} = (\mu_k)_{k \leq K}$ parameters and ‘alpha1’ corresponds to $\boldsymbol{\alpha} = (\alpha_k)_{k \leq K}$ parameters.

```

alpha = list(alpha0 = NULL,
             alpha1 = setNames(paste0("alpha_1", 1:5), paste0("yG", 1:5)))

```

These settings are critical for translating the model between the **REMixed** package and the Monolix software.

4.2 Initialization

As described in Section 2.3.6, we implement an initialization strategy that performs multiple runs of the SAEM algorithm on randomly formed small groups of biomarkers, selecting the configuration yielding the highest log-likelihood. This ensures that early parameter estimates are sufficiently close to the true values to allow stable log-likelihood computation.

```

init <- initStrat(project, alpha, ObsModel.transfo, seed = 1710)
plotInit(init)

```

In this demo, the biomarkers were randomly grouped into (G1,G4), (G2,G3), and (G5). As shown in Figure 4, the group including G1—known to be truly associated with the latent compartment—produced the highest log-likelihood across initialization runs. This provides empirical support for the effectiveness of the initialization procedure.

A comprehensive summary of the initialization results, including the estimated parameters for each group and scores, is available in Appendix D. These logs are stored in the ‘summary_init.txt’ file described in Section 3 and are automatically generated in the ‘remix’ subfolder of the Monolix project directory.

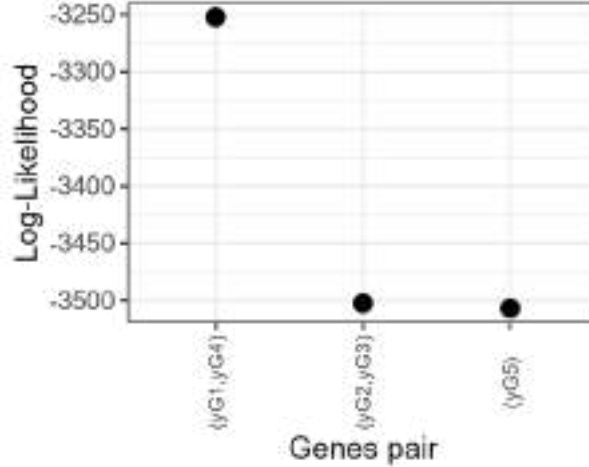


Figure 4: Log-likelihood of each group of genes tested for initialization.

4.3 Grid Search with REMixed

After initialization, we apply the REMixed algorithm to a grid of penalty values to select an optimal regularization parameter. The following command performs the grid search using 10 automatically chosen λ values. We retained 10 for illustration purposes to ensure reasonable computation time; in practice, a finer grid may be considered to refine the search. The corresponding outputs, including selection paths and BICc plots, together with the penalty parameter λ minimizing BICc, are illustrated in Figure 5 and obtained through the functions ‘plotCalibration’ and ‘plotIC’. The last function takes as an argument the information criterion to be used; the package provides the ‘BIC’, ‘AIC’, and ‘BICc’ functions by default, but user-defined criteria can also be provided.

```
resCV = cv.remix(project = project,
  dynFUN = dynFUN_demo,
  y = c(AB=1000,S=5),
  ObsModel.transfo = ObsModel.transfo,
  alpha = alpha,
  selfInit = TRUE,
  nlambda = 10,
  eps1=10**(-2),
  eps2=1)

plotCalibration(resCV, legend.position="right")
plotIC(resCV, criterion = BICc)
```

The tested grid $\Lambda = \{5.09, 10.17, \dots, 2554.29\}$ was automatically determined based on the maximal penalization value estimated internally. The optimal penalty $\lambda^* = 80.77$ was selected as the value minimizing the corrected Bayesian Information Criterion (BICc), equal to 6616.76 in this case, as described in Section 2.3.4. This criterion balances model com-

plexity and fit. In this example, two genes were selected, G1 and G2, with $\alpha_1 = 0.10$ and $\alpha_2 = 0.006$, respectively.

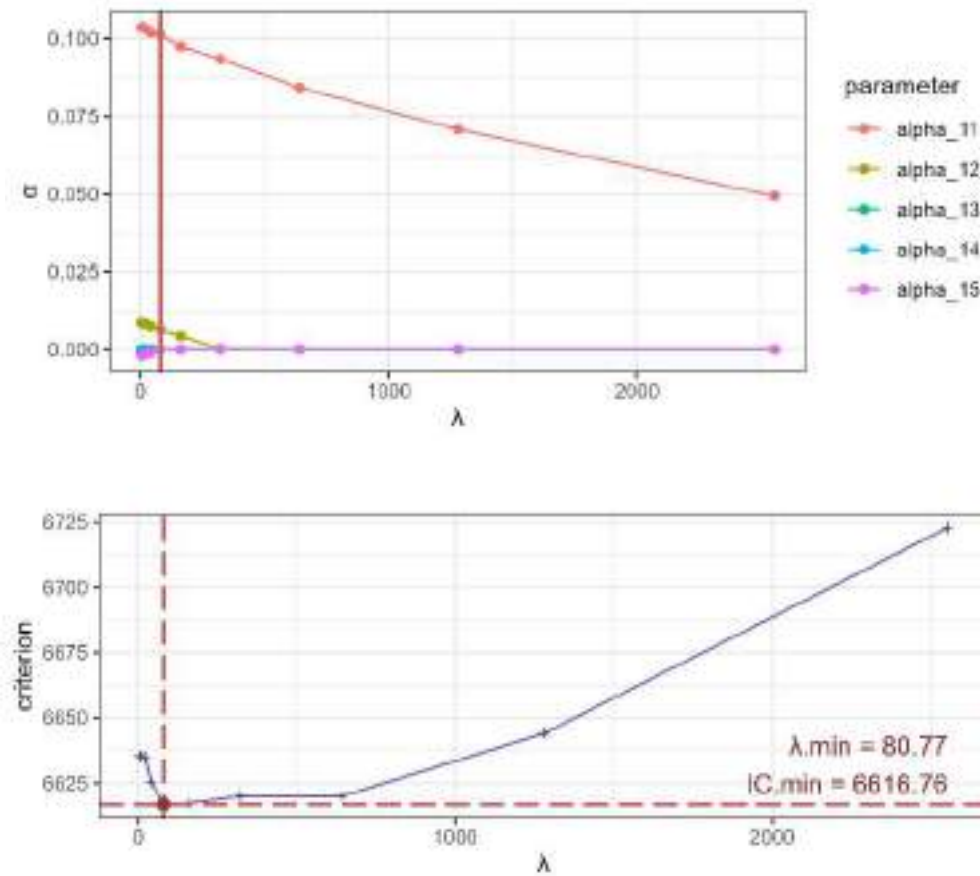


Figure 5: Calibration curve (top panel) and corrected Bayesian Information Criterion (BICc) values across the grid of penalty parameters (bottom panel) in the REMixed procedure for the demo project with 10 penalty parameters tested.

```
selected_res <- retrieveBest(resCV)
summary(selected_res)
```

```
-----
Outputs of remix algorithm from the project :
.../demoMLX/remix/demoMLX_init.mlxtran
with :
- 100 individuals;
-  $\lambda = 80.77$ .
```

• Final Estimated Parameters :

delta_S_pop	phi_S_pop	delta_AB_pop	omega_delta_S	omega_phi_S
0.25	596.32	0.02	0.16	0.61
omega_delta_AB	sigma_AB	sigma_G1	sigma_G2	sigma_G3
0.29	0.11	0.28	0.34	0.39
sigma_G4	sigma_G5			
0.36	0.32			

• Final Selected Biomarkers and Estimate:

alpha_11_pop	alpha_12_pop
0.10100	0.00638

• Final Scores :

- OFV : 6500.482
- BIC : 6564.954
- BICc : 6616.759

Executed in 2 iterations, 106 s.

All results are logged in the summary files ‘summary_cv.txt’, for the overall grid construction, and ‘summary_l.txt’, for each penalty parameter $l = 1, \dots, N_\lambda$ tested, described in Section 3, which are automatically saved in the ‘remix’ subfolder. Complete logs for this example are provided in Appendix E.

4.4 REMixed with fixed penalty parameter λ

If a specific penalty parameter λ is chosen, it is possible to run the REMixed algorithm for that fixed value to refine the final estimates. This is done using the ‘remix’ function, here with $\lambda = 80.77$:

```
res_lambda = remix(project = project,
                    dynFUN = dynFUN_demo,
                    y = c(AB=1000,S=5),
                    ObsModel.transfo = ObsModel.transfo,
                    alpha = alpha,
                    selfInit = TRUE,
                    lambda = 80.77,
                    eps1=0.01,
                    eps2=0.1)
summary(res_lambda)
```

Outputs of remix algorithm from the project :
...demoMLX_upd.mlxtran

```

with :
  - 100 individuals;
  -  $\lambda = 80.77$ .
-----

• Final Estimated Parameters :
      delta_S_pop      phi_S_pop      delta_AB_pop      omega_delta_S      omega_phi_S
          0.24          570.69          0.02          0.11          0.62
omega_delta_AB      sigma_AB      sigma_G1      sigma_G2      sigma_G3
          0.30          0.11          0.28          0.34          0.39
      sigma_G4      sigma_G5
          0.36          0.32
-----

• Final Selected Biomarkers and Estimate:
alpha_11_pop alpha_12_pop
      0.09760      0.00668
-----

• Final Scores :
  • OFV : 6499.817
  • BIC : 6564.29
  • BICc : 6616.094

Executed in 8 iterations, 352.2 s.
-----

```

```

trueValue <- read.csv(paste0(dirname(getMLXdir()),
                             "/demoSMLX/Simulation/populationParameters.txt"))

plotSAEM(res_lambda, trueValue = trueValue)
plotConvergence(res_lambda)

```

Figures 6 and 7 illustrate the convergence of the SAEM algorithm and the behavior of the log-likelihood across iterations. These are obtained with calls to ‘plotSAEM’ and ‘plotConvergence’. The argument ‘trueValue’, if known (as in simulation studies), allows displaying the true values of each parameter alongside their estimates. This step refines the parameter estimates based on the selected biomarkers and updates the Monolix project accordingly. As with previous steps, a full log is saved in the file ‘summary.txt’ and included in Appendix F.

4.5 Final Statistical Tests

To balance the shrinkage introduced by the Lasso penalty and validate the relevance of the selected biomarkers, we conduct a final run of the SAEM algorithm followed by Wald-based statistical testing, using the ‘computeFinalTest’ function.

This step recomputes the SAEM estimates without penalization on the selected subset of biomarkers and provides standard errors and 95% confidence intervals. Wald tests are then

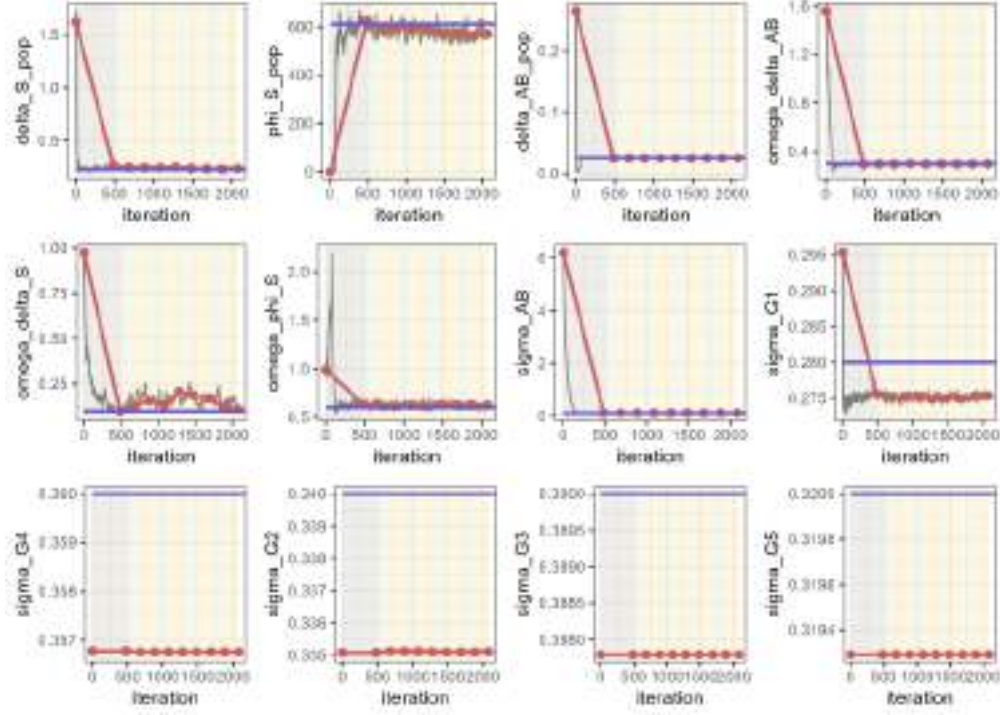


Figure 6: Evolution of parameter estimates across iterations of the SAEM algorithm within the REMixed procedure for the demo project. Initialization iterations are shown in grey, and REMixed iterations are displayed in alternating green and yellow segments.

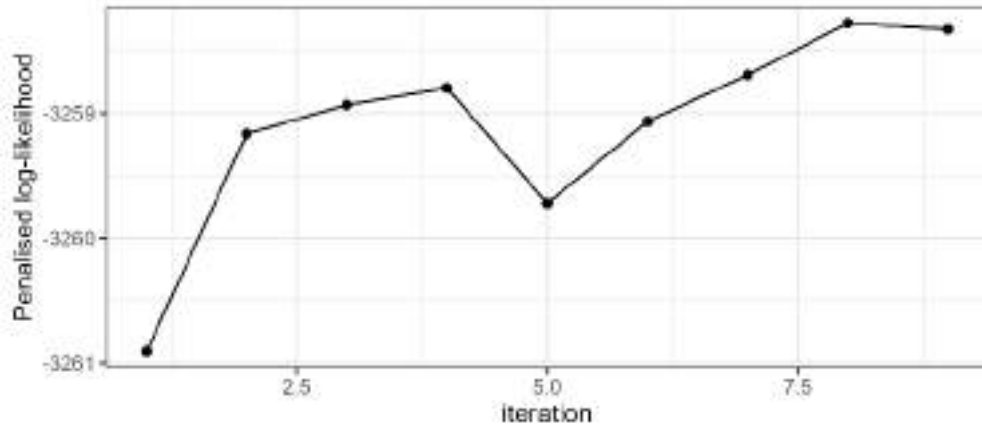


Figure 7: Trajectory of penalized log-likelihood evolution over SAEM iterations in the REMixed procedure for the demo project.

used to confirm the significance of each selected biomarker, ensuring that only those with robust support remain in the final model. If the set of selected biomarkers changes, a final estimation step is carried out using the SAEM algorithm.

In the demo project, biomarker G2 was removed by the final test, confirming the validity of the previous selection of G1 only. Parameter estimates, standard errors, confidence

intervals and p-value are reported in Appendix G, along with full logs saved in the final ‘summary.txt’ file within the ‘remix’ folder of the corresponding final project. After this test, and since the selected set had changed, a final run of the SAEM algorithm was performed, jointly estimating NLME parameters and the parameters related to biomarker G1.

```
res_final <- computeFinalTest(res_lambda, dynFUN_demo, c(AB=1000,S=5),
                             ObsModel.transfo)
```

This final validation step not only provides inferential guarantees but also prepares the model for interpretation and reporting, making it a critical component of the REMixed workflow. The final outputs can be investigated as an object from the ‘remix’ function.

```
res_final <- computeFinalTest(res_lambda, dynFUN_demo, c(AB=1000,S=5),
                             ObsModel.transfo)
```

```
summary(res_final)
```

```
-----
Outputs of remix algorithm from the project :
```

```
.../demoMLX_upd_final.mlxtran
```

```
with :
```

- 100 individuals;
- $\lambda = 382.22$;
- final SAEM computed;
- final test computed.

```
-----
• Final Estimated Parameters :
```

	EstimatedValue	se	CI_95
delta_S_pop	2.4e-01	9.5e-03	[2.2e-01;2.6e-01]
phi_S_pop	5.9e+02	4.1e+01	[5e+02;6.7e+02]
delta_AB_pop	2.5e-02	7.5e-04	[2.3e-02;2.6e-02]
omega_delta_S	1.7e-01	5e-02	[6.7e-02;2.6e-01]
omega_phi_S	6.1e-01	5e-02	[5.2e-01;7.1e-01]
omega_delta_AB	3e-01	2.2e-02	[2.5e-01;3.4e-01]
sigma_AB	1.1e-01	4.1e-03	[1.1e-01;1.2e-01]
sigma_G1	2.8e-01	4.3e-03	[2.7e-01;2.8e-01]
sigma_G2	3.4e-01	5.2e-03	[3.2e-01;3.5e-01]
sigma_G3	3.9e-01	6e-03	[3.8e-01;4e-01]
sigma_G4	3.6e-01	5.5e-03	[3.5e-01;3.7e-01]
sigma_G5	3.2e-01	4.9e-03	[3.1e-01;3.3e-01]

```
-----
• Final Selected Biomarkers and Estimate:
```

	EstimatedValue	se	CI_95
alpha_11_pop	1e-01	4.9e-03	[9.2e-02;1.1e-01]

```
-----
```


- Final Scores :
 - OFV : 6502.757
 - BIC : 6562.624
 - BICc : 6609.719

Executed in 8 iterations, 429.1 s.

5 Conclusion

In this paper, we presented **REMixed**, a new framework implemented in R for integrating large-dimensional longitudinal biomarkers into nonlinear mixed-effects models based on ODEs. This approach extends regularized regression techniques to a mechanistic modeling context, enabling simultaneous parameter estimation and biomarker selection within ODE-based mixed-effects models. Using a cyclical coordinate descent algorithm to maximize a penalized likelihood, **REMixed** dynamically links a large set of candidate biomarkers to latent biological compartments. We described the algorithm step by step and applied it to a simulated dataset, illustrating how **REMixed** can identify relevant biomarkers driving the latent processes in an ODE system while accurately estimating model parameters.

The performance of **REMixed** depends on initialization, as the convergence of SAEM can be sensitive to starting values. In practice, grouping biomarkers into small sets proved effective in balancing computational feasibility with estimation stability. Other strategies could further improve robustness, such as sequentially including biomarkers or exploiting directly observed compartments when available and sufficient for parameter estimation. Moreover, computational cost increases with both the number of biomarkers and the complexity of the mechanistic model. An initial preselection step based on prior biological knowledge or simpler models may therefore help reduce this burden.

The reliance on adaptive Gauss–Hermite quadrature for likelihood approximation can also become prohibitive when many random effects are included. Exploring alternative integration schemes, such as stochastic approximation or refined quadrature methods, could improve scalability. Similarly, inference approaches like particle filtering, variational inference, or sequential Monte Carlo may offer greater stability in challenging scenarios, but at the cost of additional implementation complexity. Finally, selection of the penalty parameter in our implementation relied on the corrected Bayesian Information Criterion (BICc), but the framework can easily accommodate alternatives such as AIC or BIC. Cross-validation could also be used to calibrate the penalty parameter and help mitigate overfitting; however, the repeated model estimations it requires were not computationally feasible in our context.

Interoperability is another important dimension. The current implementation relies on Monolix via **lixoftConnectors**, but broader compatibility with tools such as NONMEM or **saemix** would facilitate adoption across pharmacometrics, epidemiology, and immunology. Moreover, it would be interesting to integrate or compare the method with established pharmacometric software such as NONMEM, or fully R-based frameworks such as the **saemix** package.

The current implementation assumes independent, homoscedastic Gaussian errors and

constant intercepts in the biomarker model. Allowing heteroscedasticity or temporal correlation in the residuals could improve realism for complex data. For instance, introducing time-varying intercepts $\mu_k(t)$ or more flexible variance functions $\sigma_k(\cdot)$ could better capture time-dependent biomarker influences on the latent process.

At present, biomarkers are linked to a single latent compartment, but the **REMixed** algorithm could be extended to multiple latent compartments in the ODE system, allowing identification of distinct biomarker signatures associated with different biological processes, such as diverse immune cell populations. The methodology also opens perspectives for broader applications. Extensions to multi-omics data, including transcriptomics, proteomics, or metabolomics, would support integrative analyses of complex biological responses. By identifying dynamic biomarker signatures associated with latent processes, **REMixed** may contribute to advances in precision medicine.

Acknowledgment

This study was carried out in the framework of the University of Bordeaux’s France 2030 program RRI PHDS (*Public Health Data Science*). Elements of this work have been carried out as part of the SOLVE project which has received funding from the European Union’s Horizon Research and Innovation program under Grant Agreement No. 101137185. Computer time for this study was provided by the computing facilities of the PlaFRIM (Plateforme Fédérative pour la Recherche en Informatique et Mathématiques).

We thank Simulations Plus, Lixoft division for the free academic use of the MonolixSuite. We disclose that generative AI tools were used to polish English language.

References

- Caillebotte, A., Kuhn, E., and Lemler, S. (2024). Estimation and variable selection in a joint model of survival times and longitudinal outcomes with random effects.
- Delattre, M., Lavielle, M., and Poursat, M.-A. (2014). A note on bic in mixed-effects models. *Electronic Journal of Statistics*, 8.
- Delyon, B., Lavielle, M., and Moulines, E. (1999). Convergence of a stochastic approximation version of the EM algorithm. *The Annals of Statistics*, 27(1):94 – 128.
- Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 39:1–22.
- Friedman, J. H., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22.
- Gabaut, A., Thiébaut, R., Proust-Lima, C., and Prague, M. (2025). A stability-enhanced lasso approach for covariate selection in non-linear mixed effect model. *medRxiv*.
- Ganser, I., Buckeridge, D. L., Heffernan, J., Prague, M., and Thiébaut, R. (2024). Estimating the population effectiveness of interventions against covid-19 in france: A modelling study. *Epidemics*, 46:100744.
- Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97.
- Ibarra, M., Lorier, M., and Trocóniz, I. F. (2021). Pharmacometrics: Population approach. In *The ADME Encyclopedia: A Comprehensive Guide on Biopharmacy and Pharmacokinetics*, pages 1–13. Springer.
- Karline Soetaert, Thomas Petzoldt, and R. Woodrow Setzer (2010). Solving differential equations in R: Package deSolve. *Journal of Statistical Software*, 33(9):1–25.
- Kuhn, E. and Lavielle, M. (2005). Maximum likelihood estimation in nonlinear mixed effects models. *Computational Statistics & Data Analysis*, 49(4):1020–1038.
- Lavielle, M. (2014). *Mixed Effects Models for the Population Approach: Models, Tasks, Methods and Tools*. Chapman and Hall/CRC.
- Li, Y., Wang, S., Song, P. X., Wang, N., Zhou, L., and Zhu, J. (2018). Doubly regularized estimation and selection in linear mixed-effects models for high-dimensional longitudinal data. *Statistics and its Interface*, 11:721–737.
- Lixoft SAS, a Simulations Plus company (2023a). Monolix. Version 2023R1.
- Lixoft SAS, a Simulations Plus company (2023b). Simulx. Version 2023R1.

- Lowe, R., Shirley, N., Bleackley, M., Dolan, S., and Shafee, T. (2017). Transcriptomics technologies. *PLoS computational biology*, 13(5):e1005457.
- Ni, X., Zhang, D., and Zhang, H. H. (2010). Variable selection for semiparametric mixed models in longitudinal studies. *Biometrics*, 66(1):79–88.
- Pan, J. and Huang, C. (2014). Random effects selection in generalized linear mixed models via shrinkage penalty function. *Statistics and Computing*, 24:725–738.
- Park, M. Y. and Hastie, T. (2007). L1-Regularization Path Algorithm for Generalized Linear Models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 69(4):659–677.
- Pasin, C., Balelli, I., Van Effelterre, T., Bockstal, V., Solforosi, L., Prague, M., Douoguih, M., and Thiébaut, R. (2019). Dynamics of the humoral immune response to a prime–boost ebola vaccine: Quantification and sources of variation. *Journal of Virology*, 93(18):e00579–19.
- Perelson, A. S. and Ribeiro, R. M. (2018). Introduction to modeling viral infections and immunity. *Immunological reviews*, 285(1):5.
- Philipps, V., Hejblum, Boris, P., Prague, M., Commenges, D., and Proust-Lima, C. (2021). Robust and efficient optimization using a marquardt-levenberg algorithm with r package marqlevalg. *The R Journal*, 13(2):273.
- R Core Team (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- Sun, J. and Basu, S. (2024). Penalized joint models of high-dimensional longitudinal biomarkers and a survival outcome. *The Annals of Applied Statistics*, 18(2):1490 – 1505.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288.
- Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization 1.
- Zhao, P., Tang, A., and Tang, N. (2014). Double penalized variable selection procedure for partially linear models with longitudinal data. *Acta Mathematica Sinica, English Series*, 30.

6 Appendix

A Regularization step details

We denote the Hessian $\frac{\partial^2}{\partial \alpha^2} LL(\alpha, \hat{\theta})$ in $\alpha = \hat{\alpha}$ as $\frac{\partial^2}{\partial \alpha^2} LL(\alpha, \hat{\theta}) \Big|_{\hat{\alpha}} = -\hat{X} = -(\hat{x}_{lc})_{l,c \leq K}$. We suppose here that the opposite of the Hessian is definite positive. Then:

$$\begin{aligned} LL(\alpha, \hat{\theta}) &\approx LL(\hat{\alpha}, \hat{\theta}) + (\alpha - \hat{\alpha})^T \frac{\partial}{\partial \alpha} LL(\alpha, \hat{\theta}) \Big|_{\alpha=\hat{\alpha}} - \frac{1}{2} (\alpha - \hat{\alpha})^T \hat{X} (\alpha - \hat{\alpha}) \\ &= LL(\hat{\alpha}, \hat{\theta}) + \alpha^T \partial_{\alpha} LL(\alpha, \hat{\theta})|_{\hat{\alpha}} - \hat{\alpha}^T \partial_{\alpha} LL(\alpha, \hat{\theta})|_{\hat{\alpha}} \\ &\quad - \frac{1}{2} \left(\alpha^T \hat{X} \alpha + \hat{\alpha}^T \hat{X} \hat{\alpha} - \alpha^T \hat{X} \hat{\alpha} - \hat{\alpha}^T \hat{X} \alpha \right). \end{aligned}$$

We'll focus on the term depending on α , as we wish to derivate with respect to α_k for $k = 1, \dots, K$, and develop the matrix products, and isolate the term depending on a specific α_k :

$$\begin{aligned} \alpha^T \partial_{\alpha} LL(\alpha, \hat{\theta})|_{\hat{\alpha}} &= \sum_{l=1}^K \alpha_l \partial_{\alpha_l} LL(\alpha, \hat{\theta})|_{\hat{\alpha}} \\ &= \alpha_k \partial_{\alpha_k} LL(\alpha, \hat{\theta})|_{\hat{\alpha}} + \sum_{\substack{l=1 \\ l \neq k}}^K \alpha_l \partial_{\alpha_l} LL(\alpha, \hat{\theta})|_{\hat{\alpha}}, \end{aligned}$$

with

$$\alpha^T \hat{X} \alpha = \sum_{c=1}^K \sum_{l=1}^K \alpha_l \hat{x}_{lc} \alpha_c = \sum_{l,c \neq k} \alpha_l \hat{x}_{lc} \alpha_c + 2\alpha_k \sum_{\substack{l=1 \\ l \neq k}}^K \hat{x}_{lk} \alpha_l + \alpha_k^2 \hat{x}_{kk},$$

$$\alpha^T \hat{X} \hat{\alpha} = \sum_{c=1}^K \hat{\alpha}_c \left(\sum_{\substack{l=1 \\ l \neq k}}^K \alpha_l \hat{x}_{lc} + \alpha_k \hat{x}_{kc} \right) = \sum_{c=1}^K \sum_{\substack{l=1 \\ l \neq k}}^K \alpha_l \hat{x}_{lc} \hat{\alpha}_c + \alpha_k \sum_{c=1}^K \hat{x}_{kc} \hat{\alpha}_c,$$

$$\hat{\alpha}^T \hat{X} \alpha = \sum_{c=1}^K \sum_{l=1}^K \hat{\alpha}_l \hat{x}_{lc} \alpha_c = \sum_{\substack{c=1 \\ c \neq k}}^K \sum_{l=1}^K \hat{\alpha}_l \hat{x}_{lc} \alpha_c + \alpha_k \sum_{l=1}^K \hat{\alpha}_l \hat{x}_{lk}.$$

Hence

$$\begin{aligned}
LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}}) &\approx LL(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\theta}}) + \alpha_k \partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} + \sum_{\substack{l=1 \\ \neq k}}^K \alpha_l \partial_{\alpha_l} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} - \hat{\boldsymbol{\alpha}}^T \partial_{\boldsymbol{\alpha}} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} \\
&- \frac{1}{2} \left(\hat{\boldsymbol{\alpha}}^T \hat{\mathbf{X}} \hat{\boldsymbol{\alpha}} + \sum_{l, c \neq k} \alpha_l \hat{x}_{lc} \alpha_c + 2\alpha_k \sum_{\substack{l=1 \\ \neq k}}^K \hat{x}_{lk} \alpha_l + \alpha_k^2 \hat{x}_{kk} \right. \\
&\quad - \sum_{c=1}^K \sum_{\substack{l=1 \\ \neq k}}^K \alpha_l \hat{x}_{lc} \hat{\alpha}_c - \alpha_k \sum_{c=1}^K \hat{x}_{kc} \hat{\alpha}_c \\
&\quad \left. - \sum_{\substack{c=1 \\ \neq k}}^K \sum_{l=1}^K \hat{\alpha}_l \hat{x}_{lc} \alpha_c - \alpha_{k'} \sum_{l=1}^K \hat{\alpha}_l \hat{x}_{lk} \right).
\end{aligned}$$

We derivate the last formula with respect to α_k

$$\begin{aligned}
\partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}}) &= \partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} - \frac{1}{2} \left(2 \sum_{\substack{l=1 \\ \neq k}}^K \hat{x}_{lk} \alpha_l + 2\alpha_k \hat{x}_{kk} - \sum_{c=1}^K \hat{x}_{kc} \hat{\alpha}_c - \sum_{k=1}^K \hat{\alpha}_l \underbrace{\hat{x}_{lk}}_{=\hat{x}_{kl}} \right) \\
&= \partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} - \sum_{\substack{l=1 \\ \neq k}}^K \hat{x}_{lk} \alpha_l - \alpha_k \hat{x}_{kk} + \sum_{c=1}^K \hat{x}_{kc} \hat{\alpha}_c.
\end{aligned}$$

We want to find a parameter α_k so that $LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}}) - \lambda \|\boldsymbol{\alpha}\|_1$ reach a maximum for this value. We then want to solve in α_k the equation

$$\begin{aligned}
\partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}}) - \lambda \text{sign}(\alpha_k) &= 0 \\
\partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} - \sum_{\substack{l=1 \\ \neq k}}^K \hat{x}_{lk} \alpha_l - \alpha_k \hat{x}_{kk} + \sum_{c=1}^K \hat{x}_{kc} \hat{\alpha}_c - \lambda \text{sign}(\alpha_k) &= 0.
\end{aligned}$$

If $\alpha_k > 0$:

$$\begin{aligned}
\partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} - \sum_{\substack{l=1 \\ \neq k}}^K \hat{x}_{lk} \alpha_l - \alpha_k \hat{x}_{kk} + \sum_{c=1}^K \hat{x}_{kc} \hat{\alpha}_c - \lambda &= 0 \\
\alpha_k &= \frac{\partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} - \sum_{\substack{l=1 \\ \neq k}}^K \hat{x}_{lk} \alpha_l + \sum_{c=1}^K \hat{x}_{kc} \hat{\alpha}_c - \lambda}{x_{kk}}.
\end{aligned}$$

So the condition $\alpha_k > 0$ implies that $\partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} - \sum_{\substack{l=1 \\ \neq k}}^K \hat{x}_{lk} \alpha_l + \sum_{c=1}^K \hat{x}_{kc} \hat{\alpha}_c > \lambda$.

If $\alpha_k < 0$:

$$\begin{aligned} \partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} - \sum_{\substack{l=1 \\ \neq k}}^K \hat{x}_{lk} \alpha_l - \alpha_k \hat{x}_{kk} + \sum_{c=1}^K \hat{x}_{kc} \hat{\alpha}_c + \lambda &= 0 \\ \alpha_k &= \frac{\partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} - \sum_{\substack{l=1 \\ \neq k}}^K \hat{x}_{lk} \alpha_l + \sum_{c=1}^K \hat{x}_{kc} \hat{\alpha}_c + \lambda}{x_{kk}}. \end{aligned}$$

So the condition $\alpha_k < 0$ implies that $\partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} - \sum_{\substack{l=1 \\ \neq k}}^K \hat{x}_{lk} \alpha_l + \sum_{c=1}^K \hat{x}_{kc} \hat{\alpha}_c < -\lambda$.

The condition for $\left| \partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} - \sum_{\substack{l=1 \\ \neq k}}^K \hat{x}_{lk} \alpha_l + \sum_{c=1}^K \hat{x}_{kc} \hat{\alpha}_c \right| < \lambda$, implies then that α_k is set to 0.

We then denote $\left(\partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} - \sum_{\substack{l=1 \\ \neq k}}^K \hat{x}_{lk} \hat{\alpha}_l + \sum_{c=1}^K \hat{x}_{kc} \hat{\alpha}_c \right) = \left(\partial_{\alpha_k} LL(\boldsymbol{\alpha}, \hat{\boldsymbol{\theta}})|_{\hat{\boldsymbol{\alpha}}} + \hat{x}_{kk} \hat{\boldsymbol{\alpha}}_k \right) = A$ an approximation using $\hat{\boldsymbol{\alpha}}$ of the previously mentionned condition.

Solving for each coordinate α_k leads to the soft-thresholding rule:

$$\alpha_k = \begin{cases} \frac{A+\lambda}{\hat{x}_{kk}} & \text{if } A < -\lambda, \\ \frac{A-\lambda}{\hat{x}_{kk}} & \text{if } A > \lambda, \\ 0 & \text{otherwise.} \end{cases} \quad \text{with } A = \partial_{\alpha_k} LL(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\alpha}}) + \hat{x}_{kk} \hat{\alpha}_k \quad (16)$$

B Structure of the project

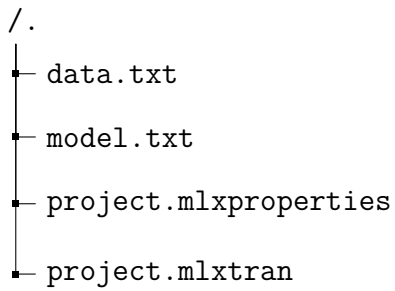


Figure 8: Structure of a Monolix project directory after creation, prior to running any estimation or simulation tasks, showing the core files required for model specification and configuration.

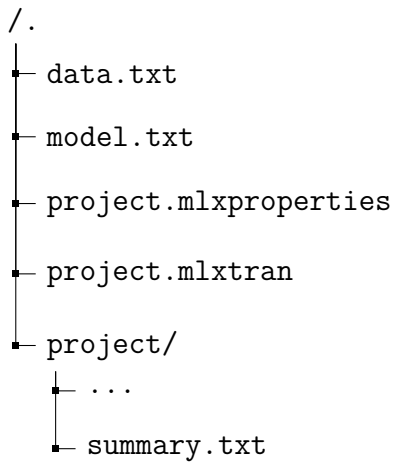


Figure 9: Structure of a Monolix project directory after task execution, illustrating the creation of a results subfolder containing output files such as `summary.txt` with estimation and diagnostic results.


```

/.
├── ...
├── project_upd.mlxtran
├── project_upd.mlxproperties
├── project_upd/...
├── project/
│   ├── ...
│   └── remix/
│       ├── Build_1/...
│       ├── ...
│       ├── project_init/...
│       ├── tmp_init/...
│       ├── Build_1.mlxtran
│       ├── Build_1.mlxproperties
│       ├── ...
│       ├── project_init.mlxtran
│       ├── project_init.mlxproperties
│       ├── summary.txt
│       ├── summary_1.txt
│       ├── ...
│       ├── summary_cv.txt
│       └── summary_init.txt

```

Figure 10: Structure of a Monolix project directory after sequential execution of `initStrat`, `remix`, and `cv.remix`, with both `unlinkTemporaryProject` and `unlinkBuildProject` set to `FALSE`. The `remix` folder contains all temporary projects and summary files generated during initialization, cross-validation, and final estimation steps.

C Mlxtran model file for demo monolix project

[LONGITUDINAL]

```
input = {delta_S,phi_S,delta_AB,alpha_11,alpha_12,alpha_13,alpha_14,alpha_15}
```

```
EQUATION:
```

```
t_0 = 0
```

```
AB_0 = 1000
```

```
S_0 = 5
```

```
ddt_S = - delta_S*S
```

```
ddt_AB = phi_S * S - delta_AB * AB
```

```
tAB = log10(AB)
```

```
G1= alpha_11 * S
```

```
G2= alpha_12 * S
```

```
G3= alpha_13 * S
```

```
G4= alpha_14 * S
```

```
G5= alpha_15 * S
```

```
OUTPUT:
```

```
output ={tAB,G1,G2,G3,G4,G5}
```

D Summary file for initialization task

```
-----
Starting Initilization by group
```

```
Group formed :
```

```
(yG1,yG4) - (yG2,yG3) - (yG5)
```

```
-----
Starting with biomarkers : yG4, yG1
```

```
Computing SAEM update for population parameters...
```

```
Estimating log-likelihood...
```

```
> Estimated Population Parameters :
```

	EstimatedValue
delta_S_pop	2.6e-01
phi_S_pop	6.3e+02
delta_AB_pop	2.5e-02
omega_delta_AB	2.9e-01
omega_delta_S	9.6e-02
omega_phi_S	6.3e-01
sigma_AB	1.1e-01
sigma_G1	2.8e-01
sigma_G2	3.4e-01
sigma_G3	3.9e-01
sigma_G4	3.6e-01
sigma_G5	3.2e-01

	EstimatedValue
alpha_11_pop	1.1e-01
alpha_12_pop	0e+00
alpha_13_pop	0e+00
alpha_14_pop	-1.1e-05
alpha_15_pop	0e+00

> Estimated logLikelihood :

LL	OFV	AIC	BIC	BICc
-3252.417	6504.830	6526.830	6555.490	6593.170

Starting with biomarkers : yG2, yG3

Computing SAEM update for population parameters...
 Estimating log-likelihood...

> Estimated Population Parameters :

	EstimatedValue
delta_S_pop	3.4e-01
phi_S_pop	7.9e+02
delta_AB_pop	2.5e-02
omega_delta_AB	3e-01
omega_delta_S	2.9e-01
omega_phi_S	5.7e-01
sigma_AB	1.1e-01
sigma_G1	3e-01
sigma_G2	3.4e-01
sigma_G3	3.9e-01
sigma_G4	3.6e-01
sigma_G5	3.2e-01

	EstimatedValue
alpha_11_pop	0e+00
alpha_12_pop	7.8e-03
alpha_13_pop	-1.7e-05
alpha_14_pop	0e+00
alpha_15_pop	0e+00

> Estimated logLikelihood :

LL	OFV	AIC	BIC	BICc
-3502.774	7005.550	7027.550	7056.210	7093.880

Starting with biomarkers : yG5

Computing SAEM update for population parameters...
 Estimating log-likelihood...

> Estimated Population Parameters :

	EstimatedValue
delta_S_pop	4e-01
phi_S_pop	9.4e+02
delta_AB_pop	2.5e-02
omega_delta_AB	3e-01
omega_delta_S	3.3e-01
omega_phi_S	5.7e-01
sigma_AB	1.2e-01
sigma_G1	3e-01
sigma_G2	3.4e-01
sigma_G3	3.9e-01
sigma_G4	3.6e-01
sigma_G5	3.2e-01

	EstimatedValue
alpha_11_pop	0e+00
alpha_12_pop	0e+00
alpha_13_pop	0e+00
alpha_14_pop	0e+00
alpha_15_pop	-4.2e-03

> Estimated logLikelihood :

LL	OFV	AIC	BIC	BICc
-3506.928	7013.860	7031.860	7055.300	7083.560

 FINAL SET SELECTED : yG4, yG1.

E Summary file for grid search

 Starting Regulatization and Estimation Algorithm

Estimation of the R.E. distribution using the initial model ...

- - - < INITIAL PARAMETERS > - - -

	EstimatedValue
delta_S_pop	2.63e-01
phi_S_pop	6.28e+02
delta_AB_pop	2.48e-02
omega_delta_AB	2.94e-01
omega_delta_S	9.57e-02

omega_phi_S	6.29e-01
sigma_AB	1.14e-01
sigma_G1	2.76e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

	EstimatedValue
alpha_11_pop	1.07e-01
alpha_12_pop	0e+00
alpha_13_pop	0e+00
alpha_14_pop	-1.09e-05
alpha_15_pop	0e+00

Computing regularization path ...

Starting algorithm n°1/10 with lambda = 5.096...

```

- - - < CRITERION > - - -
  LL : -3250.233
  BIC : 6574.149
  BICc : 6635

```

```

- - - < FINAL PARAMETERS > - - -

```

	EstimatedValue
alpha_11_pop	1.04e-01
alpha_12_pop	8.52e-03
alpha_13_pop	-7.22e-04
alpha_14_pop	0.00e+00
alpha_15_pop	-1.98e-03

	EstimatedValue
delta_S_pop	2.51e-01
phi_S_pop	6.02e+02
delta_AB_pop	2.48e-02
omega_delta_AB	2.96e-01
omega_delta_S	1.69e-01
omega_phi_S	6.14e-01
sigma_AB	1.14e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Starting algorithm n°2/10 with lambda = 10.169...

- - - < CRITERION > - - -

LL : -3250.009

BIC : 6573.7

BICc : 6635

- - - < FINAL PARAMETERS > - - -

	EstimatedValue
alpha_11_pop	1.04e-01
alpha_12_pop	8.37e-03
alpha_13_pop	-5.23e-04
alpha_14_pop	0.00e+00
alpha_15_pop	-1.85e-03
	EstimatedValue
delta_S_pop	2.49e-01
phi_S_pop	5.98e+02
delta_AB_pop	2.48e-02
omega_delta_AB	2.97e-01
omega_delta_S	1.47e-01
omega_phi_S	6.15e-01
sigma_AB	1.14e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Starting algorithm n°3/10 with lambda = 20.289...

- - - < CRITERION > - - -

LL : -3250.038

BIC : 6573.759

BICc : 6635

- - - < FINAL PARAMETERS > - - -

	EstimatedValue
alpha_11_pop	1.03e-01
alpha_12_pop	8.04e-03
alpha_13_pop	-2.14e-06
alpha_14_pop	0.00e+00
alpha_15_pop	-1.57e-03
	EstimatedValue
delta_S_pop	2.44e-01
phi_S_pop	5.87e+02

delta_AB_pop	2.48e-02
omega_delta_AB	2.98e-01
omega_delta_S	1.64e-01
omega_phi_S	6.06e-01
sigma_AB	1.15e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Starting algorithm n°4/10 with lambda = 40.483...

```

- - - < CRITERION > - - -
  LL : -3249.804
  BIC : 6568.685
  BICc : 6625

```

```

- - - < FINAL PARAMETERS > - - -

```

	EstimatedValue
alpha_11_pop	1.02e-01
alpha_12_pop	7.52e-03
alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	-1.02e-03

	EstimatedValue
delta_S_pop	2.39e-01
phi_S_pop	5.74e+02
delta_AB_pop	2.48e-02
omega_delta_AB	2.97e-01
omega_delta_S	1.43e-01
omega_phi_S	6.13e-01
sigma_AB	1.15e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Starting algorithm n°5/10 with lambda = 80.774...

```

- - - < CRITERION > - - -
  LL : -3250.241
  BIC : 6564.954
  BICc : 6617

```

- - - < FINAL PARAMETERS > - - -

	EstimatedValue
alpha_11_pop	1.01e-01
alpha_12_pop	6.38e-03
alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	0.00e+00
	EstimatedValue
delta_S_pop	2.46e-01
phi_S_pop	5.96e+02
delta_AB_pop	2.49e-02
omega_delta_AB	2.95e-01
omega_delta_S	1.58e-01
omega_phi_S	6.12e-01
sigma_AB	1.14e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Starting algorithm n°6/10 with lambda = 161.165...

- - - < CRITERION > - - -

LL : -3250.432
BIC : 6565.336
BICc : 6617

- - - < FINAL PARAMETERS > - - -

	EstimatedValue
alpha_11_pop	9.75e-02
alpha_12_pop	4.41e-03
alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	0.00e+00
	EstimatedValue
delta_S_pop	2.4e-01
phi_S_pop	5.79e+02
delta_AB_pop	2.49e-02
omega_delta_AB	2.97e-01
omega_delta_S	1.4e-01
omega_phi_S	6.22e-01
sigma_AB	1.14e-01
sigma_G1	2.75e-01

sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Starting algorithm n°7/10 with lambda = 321.566...

- - - < CRITERION > - - -

LL : -3251.821
BIC : 6568.115
BICc : 6620

- - - < FINAL PARAMETERS > - - -

	EstimatedValue
alpha_11_pop	9.35e-02
alpha_12_pop	2.54e-04
alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	0.00e+00
	EstimatedValue
delta_S_pop	2.3e-01
phi_S_pop	5.59e+02
delta_AB_pop	2.49e-02
omega_delta_AB	2.95e-01
omega_delta_S	1.16e-01
omega_phi_S	6.32e-01
sigma_AB	1.14e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Starting algorithm n°8/10 with lambda = 641.609...

- - - < CRITERION > - - -

LL : -3256.438
BIC : 6572.744
BICc : 6620

- - - < FINAL PARAMETERS > - - -

	EstimatedValue
alpha_11_pop	8.42e-02
alpha_12_pop	0.00e+00

alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	0.00e+00
	EstimatedValue
delta_S_pop	2.17e-01
phi_S_pop	5.27e+02
delta_AB_pop	2.49e-02
omega_delta_AB	2.95e-01
omega_delta_S	1.81e-01
omega_phi_S	6.18e-01
sigma_AB	1.14e-01
sigma_G1	2.76e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Starting algorithm n°9/10 with lambda = 1280.178...

```

- - - < CRITERION > - - -
  LL : -3268.485
  BIC : 6596.836
  BICc : 6644

```

```

- - - < FINAL PARAMETERS > - - -

```

	EstimatedValue
alpha_11_pop	7.08e-02
alpha_12_pop	0.00e+00
alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	0.00e+00
	EstimatedValue
delta_S_pop	2.07e-01
phi_S_pop	5.1e+02
delta_AB_pop	2.5e-02
omega_delta_AB	2.96e-01
omega_delta_S	1.21e-01
omega_phi_S	6.15e-01
sigma_AB	1.15e-01
sigma_G1	2.77e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Starting algorithm n°10/10 with lambda = 2554.292...

- - - < CRITERION > - - -

LL : -3307.982
BIC : 6675.832
BICc : 6723

- - - < FINAL PARAMETERS > - - -

	EstimatedValue
alpha_11_pop	4.94e-02
alpha_12_pop	0.00e+00
alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	0.00e+00
	EstimatedValue
delta_S_pop	1.85e-01
phi_S_pop	4.68e+02
delta_AB_pop	2.52e-02
omega_delta_AB	2.93e-01
omega_delta_S	1.22e-01
omega_phi_S	6.22e-01
sigma_AB	1.15e-01
sigma_G1	2.82e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

F Summary file for REMix algorithm

Starting Regulatization and Estimation Algorithm

- - - < INITIAL PARAMETERS > - - -

	EstimatedValue
delta_S_pop	2.63e-01
phi_S_pop	6.28e+02
delta_AB_pop	2.48e-02
omega_delta_AB	2.94e-01
omega_delta_S	9.57e-02
omega_phi_S	6.29e-01
sigma_AB	1.14e-01
sigma_G1	2.76e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01

sigma_G3	3.88e-01
sigma_G5	3.19e-01

	EstimatedValue
alpha_11_pop	1.07e-01
alpha_12_pop	0e+00
alpha_13_pop	0e+00
alpha_14_pop	-1.09e-05
alpha_15_pop	0e+00

Estimating the log-likelihood, and its derivatives, using the initial model ...

LL : -3252.263
 LL.pen : -3260.91
 time elapsed : 18.76s

 ITERATION 1

Computing taylor update for regularization parameters...

	EstimatedValue
alpha_11_pop	1.05e-01
alpha_12_pop	5.98e-03
alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	0.00e+00

Computing SAEM update for population parameters...

	EstimatedValue
delta_S_pop	2.47e-01
phi_S_pop	5.94e+02
delta_AB_pop	2.48e-02
omega_delta_AB	2.96e-01
omega_delta_S	1.3e-01
omega_phi_S	6.32e-01
sigma_AB	1.14e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Estimating penalised log-likelihood...

LL :-3250.181
 LL.pen :-3259.167

Current parameter convergence criterion : 0.005
 Current ll pen convergence criterion : 1.743
 time elapsed : 38.8s

ITERATION 2

Computing taylor update for regularization parameters...

	EstimatedValue
alpha_11_pop	1.01e-01
alpha_12_pop	6.38e-03
alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	0.00e+00

Computing SAEM update for population parameters...

	EstimatedValue
delta_S_pop	2.46e-01
phi_S_pop	5.96e+02
delta_AB_pop	2.49e-02
omega_delta_AB	2.95e-01
omega_delta_S	1.58e-01
omega_phi_S	6.12e-01
sigma_AB	1.14e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Estimating penalised log-likelihood...

LL : -3250.241
LL.pen : -3258.934

Current parameter convergence criterion : 0.001
Current ll pen convergence criterion : 0.233
time elapsed : 39.17s

ITERATION 3

Computing taylor update for regularization parameters...

	EstimatedValue
alpha_11_pop	1.01e-01
alpha_12_pop	6.43e-03
alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	0.00e+00

Computing SAEM update for population parameters...

	EstimatedValue
delta_S_pop	2.47e-01

phi_S_pop	5.91e+02
delta_AB_pop	2.48e-02
omega_delta_AB	2.99e-01
omega_delta_S	1.35e-01
omega_phi_S	6.22e-01
sigma_AB	1.13e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Estimating penalised log-likelihood...

LL :-3250.115

LL.pen :-3258.8

Current parameter convergence criterion : 0.001

Current ll pen convergence criterion : 0.134

time elapsed : 41.77s

 ITERATION 4

Computing taylor update for regularization parameters...

EstimatedValue

alpha_11_pop	1.01e-01
alpha_12_pop	6.40e-03
alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	0.00e+00

Computing SAEM update for population parameters...

EstimatedValue

delta_S_pop	2.51e-01
phi_S_pop	6e+02
delta_AB_pop	2.48e-02
omega_delta_AB	2.96e-01
omega_delta_S	2.08e-01
omega_phi_S	6.12e-01
sigma_AB	1.14e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Estimating penalised log-likelihood...

LL :-3251.027

LL.pen :-3259.722

Current parameter convergence criterion : 0.005
Current ll pen convergence criterion : 0.923
time elapsed : 41.73s

ITERATION 5

Computing taylor update for regularization parameters...

	EstimatedValue
alpha_11_pop	1.02e-01
alpha_12_pop	6.41e-03
alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	0.00e+00

Computing SAEM update for population parameters...

	EstimatedValue
delta_S_pop	2.39e-01
phi_S_pop	5.75e+02
delta_AB_pop	2.49e-02
omega_delta_AB	2.97e-01
omega_delta_S	1.86e-01
omega_phi_S	6.27e-01
sigma_AB	1.13e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Estimating penalised log-likelihood...

LL :-3250.337
LL.pen :-3259.068

Current parameter convergence criterion : 0.003
Current ll pen convergence criterion : 0.654
time elapsed : 43.41s

ITERATION 6

Computing taylor update for regularization parameters...

	EstimatedValue
alpha_11_pop	9.91e-02
alpha_12_pop	6.59e-03
alpha_13_pop	0.00e+00

alpha_14_pop 0.00e+00
alpha_15_pop 0.00e+00

Computing SAEM update for population parameters...

	EstimatedValue
delta_S_pop	2.36e-01
phi_S_pop	5.76e+02
delta_AB_pop	2.49e-02
omega_delta_AB	2.96e-01
omega_delta_S	1.71e-01
omega_phi_S	6.23e-01
sigma_AB	1.13e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Estimating penalised log-likelihood...

LL :-3250.156
LL.pen :-3258.695

Current parameter convergence criterion : 0
Current ll pen convergence criterion : 0.374
time elapsed : 41.75s

ITERATION 7

Computing taylor update for regularization parameters...

	EstimatedValue
alpha_11_pop	9.87e-02
alpha_12_pop	6.63e-03
alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	0.00e+00

Computing SAEM update for population parameters...

	EstimatedValue
delta_S_pop	2.32e-01
phi_S_pop	5.62e+02
delta_AB_pop	2.49e-02
omega_delta_AB	2.98e-01
omega_delta_S	1.24e-01
omega_phi_S	6.23e-01
sigma_AB	1.14e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01

sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Estimating penalised log-likelihood...

LL : -3249.774
LL.pen : -3258.28

Current parameter convergence criterion : 0.003
Current ll pen convergence criterion : 0.415
time elapsed : 43.35s

ITERATION 8

Computing taylor update for regularization parameters...

	EstimatedValue
alpha_11_pop	9.76e-02
alpha_12_pop	6.68e-03
alpha_13_pop	0.00e+00
alpha_14_pop	0.00e+00
alpha_15_pop	0.00e+00

Computing SAEM update for population parameters...

	EstimatedValue
delta_S_pop	2.38e-01
phi_S_pop	5.71e+02
delta_AB_pop	2.48e-02
omega_delta_AB	2.98e-01
omega_delta_S	1.11e-01
omega_phi_S	6.24e-01
sigma_AB	1.14e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

Estimating penalised log-likelihood...

LL : -3249.909
LL.pen : -3258.33

Current parameter convergence criterion : 0
Current ll pen convergence criterion : 0.05

G Summary file for final tests and estimation

Starting Final Regulatization and Estimation Algorithm Step

- - - < INITIAL PARAMETERS > - - -

	EstimatedValue
delta_S_pop	2.38e-01
phi_S_pop	5.71e+02
delta_AB_pop	2.48e-02
omega_delta_AB	2.98e-01
omega_delta_S	1.11e-01
omega_phi_S	6.24e-01
sigma_AB	1.14e-01
sigma_G1	2.75e-01
sigma_G4	3.57e-01
sigma_G2	3.35e-01
sigma_G3	3.88e-01
sigma_G5	3.19e-01

	EstimatedValue
alpha_11_pop	9.76e-02
alpha_12_pop	6.68e-03
alpha_13_pop	0e+00
alpha_14_pop	0e+00
alpha_15_pop	0e+00

Computing final SAEM...

Estimating log-likelihood...

	EstimatedValue	se	CI_95
delta_S_pop	2.32e-01	8.5e-03	[2.16e-01;2.49e-01]
phi_S_pop	5.64e+02	4.03e+01	[4.85e+02;6.43e+02]
delta_AB_pop	2.49e-02	7.45e-04	[2.35e-02;2.64e-02]
alpha_11_pop	9.91e-02	4.72e-03	[8.99e-02;1.08e-01]
alpha_12_pop	9.1e-03	5.15e-03	[-9.87e-04;1.92e-02]
alpha_13_pop	0e+00		
alpha_14_pop	0e+00		
alpha_15_pop	0e+00		
omega_delta_S	1.44e-01	3.28e-02	[7.98e-02;2.08e-01]
omega_phi_S	6.27e-01	5.12e-02	[5.26e-01;7.27e-01]
omega_delta_AB	2.95e-01	2.14e-02	[2.53e-01;3.37e-01]
sigma_AB	1.14e-01	4.01e-03	[1.06e-01;1.22e-01]
sigma_G1	2.75e-01	4.26e-03	[2.67e-01;2.84e-01]
sigma_G2	3.35e-01	5.17e-03	[3.25e-01;3.45e-01]
sigma_G3	3.88e-01	5.98e-03	[3.76e-01;4e-01]
sigma_G4	3.57e-01	5.5e-03	[3.46e-01;3.68e-01]
sigma_G5	3.19e-01	4.93e-03	[3.1e-01;3.29e-01]

```

- - - < CRITERION > - - -
  LL : -3249.806
  BIC : 6564.085
  BICc : 6615.889

```

Computing Wald test (with null hypothesis $\alpha_1=0$)...

	EstimatedValue	CI_95	stat.test	p.value
alpha_11_pop	9.91e-02	[8.99e-02;1.08e-01]	21.011	0 <0.05
alpha_12_pop	9.1e-03	[-9.87e-04;1.92e-02]	1.768	0.077
alpha_13_pop	0e+00			
alpha_14_pop	0e+00			
alpha_15_pop	0e+00			

```

  Setting parameters to 0...
(re)Computing final SAEM...
Estimating log-likelihood...

```

```

- - - < FINAL PARAMETERS > - - -

```

	EstimatedValue	se	CI_95
delta_S_pop	2.43e-01	9.5e-03	[2.24e-01;2.62e-01]
phi_S_pop	5.86e+02	4.14e+01	[5.05e+02;6.67e+02]
delta_AB_pop	2.49e-02	7.47e-04	[2.34e-02;2.63e-02]
alpha_11_pop	1.02e-01	4.92e-03	[9.24e-02;1.12e-01]
alpha_12_pop	0e+00		
alpha_13_pop	0e+00		
alpha_14_pop	0e+00		
alpha_15_pop	0e+00		
omega_delta_S	1.66e-01	5.05e-02	[6.66e-02;2.65e-01]
omega_phi_S	6.14e-01	5.01e-02	[5.15e-01;7.12e-01]
omega_delta_AB	2.96e-01	2.16e-02	[2.53e-01;3.38e-01]
sigma_AB	1.14e-01	4.08e-03	[1.06e-01;1.22e-01]
sigma_G1	2.75e-01	4.26e-03	[2.67e-01;2.83e-01]
sigma_G2	3.35e-01	5.17e-03	[3.25e-01;3.45e-01]
sigma_G3	3.88e-01	5.98e-03	[3.76e-01;4e-01]
sigma_G4	3.57e-01	5.5e-03	[3.46e-01;3.68e-01]
sigma_G5	3.19e-01	4.93e-03	[3.1e-01;3.29e-01]

```

- - - < CRITERION > - - -
  LL : -3251.378
  BIC : 6562.624
  BICc : 6609.719

```