MULTILEVEL ANALYSIS IN R WITH A BROKEN-LINE EXAMPLE

JAN DE LEEUW

ABSTRACT. Code in R is provided for multilevel slopes-as-outcomes analysis. The current implementation minimizes either the full negative log-likelihood or the restricted negative log-likelihood using the scoring method. We use a simple repeated measures example to illustrate the analysis.

1. GENERALITIES

1.1. **Slopes-as-Outcomes Model.** We have m groups, and in each group we have an $n_j \times p$ matrix X_j of regressors and a vector y_j of n_j outcomes. The model we use, in the notation of De Leeuw and Meijer [2008][1], is

$$\underline{y}_j = X_j \underline{b}_j + \underline{\varepsilon}_j, \tag{1a}$$

$$\underline{b}_j = Z_j \gamma + \underline{\delta}_j, \tag{1b}$$

where the Z_j are $p \times m$ matrices. The disturbances $(\underline{\varepsilon}_j, \underline{\delta}_j)$ are centered, uncorrelated over different j, and have

$$\mathbf{E}\left(\begin{bmatrix}\underline{\varepsilon}_j \\ \underline{\delta}_j\end{bmatrix}\begin{bmatrix}\underline{\varepsilon}_j' & \underline{\delta}_j'\end{bmatrix}\right) = \begin{bmatrix}\sigma^2 I_j & 0 \\ 0 & \Omega\end{bmatrix}. \tag{1c}$$

Equation (1a) is the *first-level* regression model, Equation (1b) the *second level* regression model. Equation (1b) explains why we call the model *slopes-as-outcomes*. The vectors with random coefficients $\underline{B}_j$ have p elements, while there are m fixed coefficients in γ. I_j is the identity matrix of order j and Ω is a symmetric positive semi-definite matrix of order p.

Date: August 12, 2008 — 14h 30min — Typeset in TIMES ROMAN.

[1]We refer to De Leeuw and Meijer [2008] throughout, *not* because it is the first, the only, or the best reference, but because it brings together formulas and references is a single convenient place. And we use the same notation, both in the text and wherever feasible in the program.

We can also summarize equations (1) in a single regression model with correlated errors, by substituting (1b) into (1a). This gives

$$\underline{y}_j = X_j Z_j \gamma + X_j \underline{\delta}_j + \underline{\varepsilon}_j, \tag{2a}$$

which implies

$$\mathbf{E}(\underline{y}_j) = X_j Z_j \gamma, \tag{2b}$$

$$\mathbf{V}(\underline{y}_j) = X_j \Omega X_j' + \sigma^2 I. \tag{2c}$$

We parametrize Ω as a linear combination of G known matrices C_g of order p, i.e.

$$\Omega = \sum_{g=1}^{G} \xi_g C_g. \tag{3}$$

Thus the unknown parameters we have to estimate are γ, σ^2, and ξ.

1.2. **Initial Estimates of the Regression Coefficients.** Least squares estimates of γ can be computed by minimizing

$$\sigma_{OS}(\gamma) = \sum_{j=1}^{m} (y_j - X_j Z_j \gamma)'(y_j - X_j Z_j \gamma) \tag{4a}$$

We can always write y_j in the form $y_j = X_j b_j + e_j$, where b_j is any least squares estimate, i.e. any b_j such that $X_j'(y_j - X_j b_j) = X_j' e_j = 0$. Note that the least squares estimate is unique if and only if X_j has rank p. This means that

$$\sigma_{OS}(\gamma) = \sum_{j=1}^{m} (b_j - Z_j \gamma)' X_j' X_j (b_j - Z_j \gamma), \tag{4b}$$

and we can compute the estimate of γ in two steps: first compute any least squares estimates b_j and then minimize (4b). This gives

$$\hat{\gamma}_{OS} = \left(\sum_{j=1}^{m} Z_j' X_j' X_j Z_j \right)^{-1} \sum_{j=1}^{m} Z_j' X_j' X_j b_j.$$

Because $X_j b_j$ is the same for all least squares estimates, it does not matter which one we use. The least squares estimate always exists and is an unbiased estimate of γ.

In De Leeuw and Meijer [2008, p 24–26] estimate (4b) is referred to as the one-step estimate, for historical reasons. There is also a two-step estimate, given by

$$\hat{\gamma}_{TS} = \left(\sum_{j=1}^{m} Z_j' Z_j \right)^{-1} \sum_{j=1}^{m} Z_j' b_j.$$

Note that, in the singular case, $\hat{\gamma}_{TS}$ will depend on the choice of the least squares estimate b_j. Our program implements both options, but $\hat{\gamma}_{OS}$ is the default.

Clearly the optimum value of the least squares loss functions can also be used to provide an unbiased estimate of σ^2 [De Leeuw and Meijer, 2008, p. 28]. Because of planned generalizations, and because we want to deal with the possibility that $n_j = p$ for all j, we develop a more elaborate estimate in Subsection 1.5.

1.3. **Loss Function.** The loss function we minimize to get final parameter estimates in the current version of the program is, except for some irrelevant constants, the negative multinormal log-likelihood (FIML). For group j this is

$$\mathscr{L}_j(\gamma,\sigma^2,\Omega) = \log\mathbf{det}(V_j) + (y_j - X_jZ_j\gamma)'V_j^{-1}(y_j - X_jZ_j\gamma), \tag{5a}$$

where

$$V_j = X_j\Omega X_j' + \sigma^2 I. \tag{5b}$$

This can be rewritten in the computationally more efficient form [De Leeuw and Meijer, 2008, p. 21]

$$\mathscr{L}_j = (n_j - p)(\log\sigma^2 - \frac{s_j^2}{\sigma^2}) + \log\mathbf{det}(W_j) + (b_j - Z_j\gamma)'W_j^{-1}(b_j - Z_j\gamma), \tag{6a}$$

where

$$W_j = \Omega + \sigma^2(X_j'X_j)^{-1}, \tag{6b}$$

and where

$$b_j = (X_j'X_j)^{-1}X_j'y_j, \tag{6c}$$

$$s_j^2 = \frac{1}{n_j - p}(y_j - X_jb_j)'(y_j - X_jb_j), \tag{6d}$$

are the ordinary least squares regression coefficients and mean squared error. Of course the FIML loss over all groups is simply

$$\mathscr{L}_F(\gamma,\sigma^2,\Omega) = \sum_{j=1}^{m}\mathscr{L}_j(\gamma,\sigma^2,\Omega).$$

Alternatively, we can use the restricted or residual (REML) loss function. This is defined as

$$\mathscr{L}_R(\sigma^2,\Omega) = \min_{\gamma}\mathscr{L}_F(\gamma,\sigma^2,\Omega) + \log\mathbf{det}(\sum_{j=1}^{m}Z_j'W_j^{-1}Z_j). \tag{7}$$

1.4. **Reparametrization.** If Ω is positive semidefinite and $\sigma^2 > 0$ then it follows that V_j is non-singular. But W_j can still be singular if $X_j'X_j$ is singular, and the null-space of matrices Ω and $X_j'X_j$ have a non-trivial intersection.

It is not uncommon that the matrices X_j are singular, because there are too few observations in the corresponding group. In the case the formulas Subsection 1.3 do not apply any more, and we must to be more careful. Note that (6d) also does not apply when $p = n_j$, no matter if X_j is singular or non-singular.

We avoid the problem with singularity by reparametrizing the problem using the QR-decomposition $X_j = Q_jR_j$, where Q_j is $n_j \times r_j$ and satisfies $Q_j'Q_j = I$, while R_j is $r_j \times p$, and upper triangular. Here $r_j = \mathbf{rank}(X_j) \leq \min(n_j, p)$. Then

$$\underline{y}_j = \tilde{X}_j\underline{\tilde{b}}_j + \underline{\varepsilon}_j, \tag{8a}$$

$$\underline{\tilde{b}}_j = \tilde{Z}_j\gamma + \underline{\tilde{\delta}}_j, \tag{8b}$$

with $\tilde{X}_j = Q_j$, $\underline{\tilde{b}}_j = R_j\underline{b}_j$, $\tilde{Z}_j = R_jZ_j$, and $\underline{\tilde{\delta}_j} = R_j\underline{\delta}_j$. Thus

$$\mathbf{E}\left(\begin{bmatrix}\underline{\varepsilon}_j\\ \underline{\tilde{\delta}}_j\end{bmatrix}\begin{bmatrix}\underline{\varepsilon}_j' & \underline{\tilde{\delta}}_j'\end{bmatrix}\right) = \begin{bmatrix}\sigma^2 I & 0\\ 0 & \tilde{\Omega}_j\end{bmatrix}. \tag{8c}$$

with

$$\tilde{\Omega}_j = R_j\Omega R_j' = \sum_{g=1}^{G}\xi_g R_jC_gR_j' = \sum_{g=1}^{G}\xi_g\tilde{C}_{jg}.$$

We can now apply our likelihood functions (and our algorithm) to this new reparametrized system (8). Note that the parameters are exactly the same as in the old parametrization, but the new first-level regressors are orthonormal, and there is a different Ω_j for each j.

1.5. **Initial Estimate of the Variance Components.** Define $\underline{\hat{b}}_j = \tilde{X}_j'\underline{y}_j$ and

$$\underline{H}_j = (\underline{\hat{b}}_j - \tilde{Z}_j\gamma)(\underline{\hat{b}}_j - \tilde{Z}_j\gamma)'$$

Then $b_j - Z_j\gamma = \underline{\tilde{\delta}}_j + \tilde{X}_j'\underline{\varepsilon}_j$ and

$$\mathbf{E}(\underline{H}_j) = \sum_{g=1}^{G}\xi_g\tilde{C}_{jg} + \sigma^2 I.$$

Of course γ is unknown, but we have the least squares estimate $\hat{\gamma}$, which is consistent under quite general conditions. This suggest to minimize the linear least squares loss function

$$\sigma(\xi,\sigma^2)=\sum_{j=1}^{m}\mathbf{tr}\;(H_j-\sum_{g=1}^{G}\xi_g\tilde{C}_{jg}-\sigma^2 I)^2,$$

where

$$H_j=(b_j-\tilde{Z}_j\hat{\gamma})(b_j-\tilde{Z}_j\hat{\gamma})', \tag{9}$$

and $b_j=\tilde{X}_j'y_j$.

1.6. **Algorithm.** Our algorithm uses the *method of scoring* to minimize the FIML or REML loss function. More precisely, we compute FIML estimates by minimizing $\mathscr{L}_F(\gamma,\sigma^2,\xi)$ and we compute REML estimates by minimizing $\mathscr{L}_R(\gamma,\sigma^2,\xi)=\mathscr{L}_(\gamma,\sigma^2,\xi)+\log\mathbf{det}(\sum_{j=1}^{m}\tilde{Z}_j'\tilde{W}_j^{-1}\tilde{Z}_j)$.

Suppose θ is the current parameter vector with the $m+G+1$ elements (γ,σ^2,ξ), $g(\theta)$ is the value of the partials, and $H(\theta)$ is the expected value of the matrix of second derivatives. The update formula for scoring is

$$\theta^{(k+1)}=\theta^{(k)}-H(\theta^{(k)})^{-1}g(\theta^{(k)})$$

Formulas for the necessary first derivatives and for the expected values of the second derivatives are in De Leeuw and Meijer [2008, p. 33-39]. We adapt them here to our context and simplify them slightly.

Matters can be simplified considerably, by observing that the expected values of the mixed second partials of γ and the variance components σ^2 and ξ are zero. Thus

$$\gamma^{(k+1)}=\gamma^{(k)}-\left[\left.\frac{\partial^2\mathscr{L}}{\partial\gamma\partial\gamma}\right|_{\gamma=\gamma^{(k)}}\right]^{-1}\left.\frac{\partial\mathscr{L}}{\partial\gamma}\right|_{\gamma=\gamma^{(k)}}.$$

Since

$$\left.\frac{\partial\mathscr{L}}{\partial\gamma}\right|_{\gamma=\gamma^{(k)}}=-2\sum_{j=1}^{m}\tilde{Z}_j'\tilde{W}_j^{-1}(\tilde{b}_j-\tilde{Z}_j\gamma^{(k)}),$$

$$\left.\frac{\partial^2\mathscr{L}}{\partial\gamma\partial\gamma}\right|_{\gamma=\gamma^{(k)}}=2\sum_{j=1}^{m}\tilde{Z}_j'\tilde{W}_j^{-1}\tilde{Z}_j,$$

we see that, both for FIML and REML,

$$\gamma^{(k+1)}=\left[\sum_{j=1}^{m}\tilde{Z}_j'\tilde{W}_j^{-1}\tilde{Z}_j\right]^{-1}\sum_{j=1}^{m}\tilde{Z}_j'\tilde{W}_j^{-1}\tilde{b}_j. \tag{10}$$

Thus in each scoring iteration the update of $\gamma^{(k)}$ is the weighted least squares estimate of γ using the current variance components as weights. It does not depend on the current value of γ.

To update the variance components for FIML we have, using H_j from Equation (9),

$$\begin{aligned}
\frac{\partial\mathscr{L}_F}{\partial\sigma^2} &= -\sum_{j=1}^m \left\{(n_j-r_j)\left(\log\sigma^2 - \frac{s_j^2}{(\sigma^2)^2}\right) - \mathbf{tr}\,\tilde{W}_j^{-1}(H_j-\tilde{W}_j)\tilde{W}_j^{-1}\right\},\\
\frac{\partial\mathscr{L}_F}{\partial\xi_g} &= -\sum_{j=1}^m \mathbf{tr}\,\tilde{W}_j^{-1}(H_j-\tilde{W}_j)\tilde{W}_j^{-1}\tilde{C}_{jg},
\end{aligned}$$

and for the expected values of the second derivatives

$$\begin{aligned}
\mathbf{E}\left[\frac{\partial^2\mathscr{L}_F}{\partial\sigma^2\partial\sigma^2}\right] &= \sum_{j=1}^m \left\{\frac{n_j-r_j}{(\sigma^2)^2} + \mathbf{tr}\,\tilde{W}_j^{-2}\right\},\\
\mathbf{E}\left[\frac{\partial^2\mathscr{L}_F}{\partial\sigma^2\partial\xi_g}\right] &= \sum_{j=1}^m \tilde{W}_j^{-1}\tilde{C}_{jg}\tilde{W}_j^{-1},\\
\mathbf{E}\left[\frac{\partial^2\mathscr{L}_F}{\partial\xi_g\partial\xi_h}\right] &= \sum_{j=1}^m \tilde{W}_j^{-1}\tilde{C}_{jg}\tilde{W}_j^{-1}\tilde{C}_{jh}.
\end{aligned}$$

For REML we have

$$\begin{aligned}
\frac{\partial\mathscr{L}_R}{\partial\sigma^2} &= \frac{\partial\mathscr{L}_F}{\partial\sigma^2} - \mathbf{tr}\,A\Lambda,\\
\frac{\partial\mathscr{L}_R}{\partial\xi_g} &= \frac{\partial\mathscr{L}_F}{\partial\xi_g} - \mathbf{tr}\,A\Psi_g,
\end{aligned}$$

where

$$A = \left(\sum_{j=1}^m \tilde{Z}_j'\tilde{W}_j^{-1}\tilde{Z}_j\right)^{-1},$$

and

$$\begin{aligned}
\Lambda &= \sum_{j=1}^m \tilde{Z}_j'\tilde{W}_j^{-2}\tilde{Z}_j,\\
\Psi_g &= \sum_{j=1}^m \tilde{Z}_j'\tilde{W}_j^{-1}\tilde{C}_{jg}\tilde{W}_j^{-1}\tilde{Z}_j.
\end{aligned}$$

Also

$$\mathbf{E}\left[\frac{\partial^2\mathscr{L}_R}{\partial\sigma^2\partial\sigma^2}\right]=\mathbf{E}\left[\frac{\partial^2\mathscr{L}_F}{\partial\sigma^2\partial\sigma^2}\right]-\mathbf{tr}\ \Lambda A\Lambda A,$$
$$\mathbf{E}\left[\frac{\partial^2\mathscr{L}_R}{\partial\sigma^2\partial\xi_g}\right]=\mathbf{E}\left[\frac{\partial^2\mathscr{L}_F}{\partial\sigma^2\partial\xi_g}\right]-\mathbf{tr}\ \Lambda A\Psi_g A,$$
$$\mathbf{E}\left[\frac{\partial^2\mathscr{L}_R}{\partial\xi_g\partial\xi_h}\right]=\mathbf{E}\left[\frac{\partial^2\mathscr{L}_F}{\partial\xi_g\partial\xi_h}\right]-\mathbf{tr}\ \Psi_g A\Psi_h A,$$

1.7. **Post-processing.** If we have estimates of the parameters, we can use these to compute estimates of the regression coefficients and the predicted values, and we can use the expected values of the second derivatives associated with the scoring method to compute standard errors.

Other quantities that are often of interest are the best linear unbiased estimates of the random regression coefficients (the means of the conditional distribution of the regression coefficients given the data). These are [De Leeuw and Meijer, 2008, p. 26-28]

$$\hat{b}_j=\tilde{\Omega}_j\tilde{W}_j^{-1}b_j+(I-\tilde{\Omega}_j\tilde{W}_j^{-1})\tilde{Z}_j\gamma,$$

i.e. they are a matrix weighted mean of the ordinary least squares regression coefficients b_j and the maximum likelihood estimates $\tilde{Z}_j\gamma$. The $\hat{b}_j$ are also known as the *shrinkage estimates*.

1.8. **Current `R` Implementation.** The complete `R` code for the multilevel function is in Section A.1. In our implementation we use various data structures implemented as lists of lists. It would be better, in many respects, to use `S3` or `S4` objects, but that is on the agenda of future improvements. The first data structure is `allData`, which is a list of m lists. In list j we store X_j, Z_j and y_j, where the y_j may have missing data. `allData` is the input to the `multilevel` program.

The data are used to compute `allMulti`, which is another list of m lists. In these we store copies of X_j, Z_j and y_j, but cleaned up. Missing data are eliminated, together with the corresponding rows of the X_j. If the resulting X_j have zero columns, then these are elimated as well, together with the corresponding rows of the Z_j. In addition `allMulti` stores $X_j'X_j$, $(X_j'X_j)^{-1}$, $X_j'y_j$, b_j, $y_j-X_jb_j$, and s_j^2. One possible improvement of the algorithm is to be a bit less generous with reserving storage for local data.

Two more lists are needed to start the iterations. `omStruc` is a list with the matrices C_g, while `parStruc` holds the current copies of the parameter estimatesγ, σ^2, and ξ. The program `makeIniEst` computes initial estimates of γ and σ^2 by either the one-step or the two-step ordinary least squares method discussed in Subsection 1.2. Initial consistent estimates of ξ are computed with the method similar in Subsection 1.5.

The program `omMake` makes Ω from the C_g. In a future version there will be a similar program `sgMake`, which will create $\Sigma = \mathbf{E}(\underline{\varepsilon}_j\underline{\varepsilon}_j')$ from a number of parameters θ, with options for example to handle auto-regressive error structures.

During the iterations auxilary quantities such as the negative log-likelihood, the first derivatives, and the expected values of the second derivatives are stored in a list `auxStruct`. If the `multilevel` program is called with `verbose=TRUE` then intermediate function values and gradient norms for each iteration are printed out.

After convergence we fill another list of lists with post-processing results. `allPost` stores, for each group, the least squares estimate b_j, the maximum likelihood estimate $Z_j\gamma$, and the BLUP estimates $\hat{b}_j$ of the regression coefficients. For each of these three different estimates of the regression coefficients we also compute the predicted values. There is enough information available to also easily compute the standard errors of all these parameter estimates, regression coefficients, and predicted values, because `multilevel` returns a list with `parStruc`, `auxStruc`, `allPost`, and `allMulti`.

The program `makePredY` is used for repeated measure data, in which groups are individuals, with measurements on a single variable at different time points. Once we have computed our estimates, we can plot the predicted values as a continuous curve, with continuous confidence bounds.

2. Example

In the study we are interested in here, conduction velocities of eleven specimens are measured in a ischemic and non-ischemic area. Measurements are made at times 0,5,7, and 10 minutes during ischemia, and at times 11,15, and 20 minutes during reperfusion. Thus there are 22 groups of seven correlated observations each.

We assume that the regression of the response on time decreases linearly during ischemia from baseline time $t = 0$ to $t = 10$, and then increases linearly during reperfusion time to $t = 20$. In a non-ischemic area the measurements will be more or less constant over time. Thus in both ischemic and non-ischemic areas the regression function can be modeled as a continuous linear spline, or broken line, with a single knot at $t = 10$. Three such possible regression lines are shown in Figure 1.

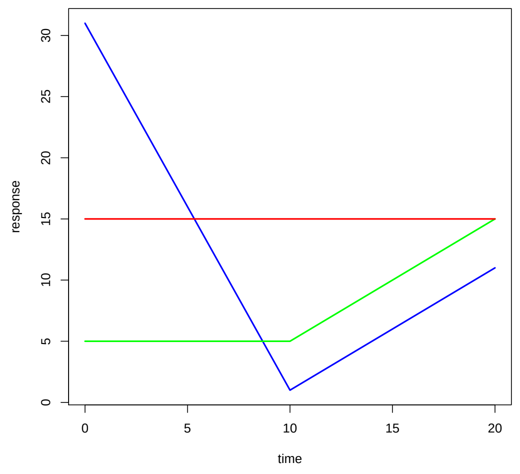

FIGURE 1.

The design matrix X_j at the seven time points is the same for all 22 groups.

$$X = \begin{bmatrix} -10 & 0 & 1 \\ -5 & 0 & 1 \\ -3 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 5 & 1 \\ 0 & 10 & 1 \end{bmatrix}$$

There will be one vector of three regression coefficients for the ischemic areas and another vector of three regression coefficients for the non-ischemic areas. In other words

$$\underline{y}_i = \begin{cases} X\beta_0 + \underline{\varepsilon}_i & \text{if } i \text{ is even,} \\ X\beta_1 + \underline{\varepsilon}_i & \text{if } i \text{ is odd,} \end{cases}$$

where we suppose the even indices correspond with non-ischemic areas and the subsequent odd indices with ischemic areas in the same specimen. The $\underline{\varepsilon}_i$ are 22 vectors of random disturbances, which we assume to be independent with mean zero and dispersion $\sigma^2 I$). This model can be fitted simply by linear regression.

There are some reasons why we can perhaps do better. Subsequent measurements on the same specimen may be correlated. This can be modeled by using correlated disturbances, and one way to get correlated disturbances is by using a slopes-as-outcomes model. The model becomes

$$\underline{y}_i = X\underline{b}_i + \underline{\varepsilon}_i,$$

where

$$\underline{b}_i = \begin{cases} \beta_0 + \underline{\delta}_i & \text{if } i \text{ is even,} \\ \beta_1 + \underline{\delta}_i & \text{if } i \text{ is odd.} \end{cases}$$

We suppose, as before, that the $\underline{\varepsilon}_i$ are uncorrelated with mean zero and dispersion $\sigma^2 I$, while the $\underline{\delta}_i$ are uncorrelated with mean zero. The dispersion matrix of the $\underline{\delta}_i$, and thus of the random regression coefficients $\underline{b}_i$, is a 3×3 matrix Ω. Note that

$$\underline{y}_i = \begin{cases} X\beta_0 + X\underline{\delta}_i + \underline{\varepsilon}_i & \text{if } i \text{ is even,} \\ X\beta_1 + X\underline{\delta}_i + \underline{\varepsilon}_i & \text{if } i \text{ is odd,} \end{cases}$$

and thus the dispersion matrix of $\underline{y}_i$ is $X\Omega X' + \sigma^2 I$. In the special case of a random intercept model, only element $(3,3)$ of Ω is non-zero. In that case the dispersion matrix of $\underline{y}_i$ is $\omega^2 E + \sigma^2 I$, where E is a matrix filled with ones. We call $\rho = \frac{\omega^2}{\omega^2+\sigma^2}$ the *intraclass correlation*, and it gives us a measure of the interdependence of the residuals within a sample.

An additional complication in the experiment we analyze is that for some samples the time points with $t > 10$ minutes (the reperfusion times) are missing. This leads to the more general model

$$\underline{y}_i = X_i\underline{b}_i + \underline{\varepsilon}_i,$$

where X_i can be either the previous X with seven rows, or it can be

$$X_i = \begin{bmatrix} -10 & 0 & 1 \\ -5 & 0 & 1 \\ -3 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Note that in this last case the complete second column of X_i is zero, which means we are just fitting a straight line to the four time points during ischemia.

Several restrictions on the second-level regression coefficients could be of interest. The first is that β_0 and β_1 are identical, i.e. the regression in ischemic and non-ischemic areas is the same. There are also other hypothesis that may make some sense. The non-ischemic lines can be horizontal, which means that β_0 has its first two elements equal to zero. We could also assume, in addition, the ischemic lines decrease to zero at $t = 10$, which means that the third element of β_1 is zero, and finally that reperfusion after 10 minutes gets back to base level, which means the first two elements of both β_0 and β_1 are equal. And then we can combine these various restrictions of β to define even more restrictive hypotheses. Since the number of observations is quite small, we generally want to keep the number of parameters small as well.

To formulate the model as a slopes-as-outcomes model, of which our simple broken line is a special case, we observe that the regression defining the random coefficients can be written as

$$\underline{b}_i = Z_i\gamma + \underline{\delta}_i,$$

where

$$\gamma = \begin{bmatrix} \beta_0 \\ \beta_i \end{bmatrix},$$

and

$$Z_i = \begin{cases} \begin{bmatrix} I & \mid & 0 \end{bmatrix} & \text{if } i \text{ is even,} \\ \begin{bmatrix} 0 & \mid & I \end{bmatrix} & \text{if } i \text{ is odd.} \end{cases}$$

The broken line results are in Figure 2 for FIML and in Figure 3 for REML, using a simple random intercept model for Ω. The code to run the examples and generate the figures is in section A.3.

The FIML and REML solutions are virtually indistinguishable. It turns out that allowing for random slopes does not really make a difference (slope variances are estimated to be very small and negative). We plotted the broken-line estimates in red for ischemic and in blue for non-ischemic areas, using the maximum likelihood estimates $X_jZ_j\gamma$. The green lines in the plot are plus or minus two standard deviations around the predictions. Data points for non-ischemic areas are in blue, data points for ischemic areas in red.

In Figure 4 we show the solution for the less restrictive case in which the two curve pieces can be quadratics. This improves the fit slightly, because the quadratic can get closer to the data points at $t = 10$ and allows for fact that the ischemic growth curve has two concave pieces. Figure 5 has the same quadratic solution, but now drawn as a continuous curve with continuous confidence bands.

We have also repeated the FIML analysis with the restrictions that the broken line for non-ischemic areas is horizontal, while the broken line for ischemic areas equals zero at $t = 10$. The solution is plotted in Figure 6. We see smaller confidence intervals, but especially for ischemic areas the restrictions do not fit the data very well.

Finally, in Figure 7, there are no functional restrictions on the form of the curve, which basically amounts to choosing the X_j to be square and non-singular. The model is, essentially,

$$\mathbf{E}(\underline{y}_i) = \begin{cases} \beta_0 & \text{if } i \text{ is even,} \\ \beta_1 & \text{if } i \text{ is odd,} \end{cases}$$

with

$$\mathbf{V}(\underline{y}_i) = \omega^2 E + \sigma^2 I.$$

Of course this should be modified slighty for imcomplete observations. This last solution seems to be most satisfactory for these data.

We compare the fit of the four models in Table 1.

	$\mathscr{L}_F$	# par	σ^2	ω^2	ρ
restricted broken line	-1007.622	5	0.000200	0.000441	0.688
broken line	-1028.057	8	0.000184	0.000269	0.594
broken quadratic	-1070.592	12	0.000129	0.000278	0.683
no curve	-1095.919	16	0.000104	0.001976	0.950

TABLE 1.

Increasing the number of parameters generally gives a significant improvement. Note that in all analyses, especially in the last one which imposes no functional form, the intraclass correlation ρ is very high, indicating a strong dependence in time.

References

J. De Leeuw and E. Meijer. Introduction to Multilevel Analysis. In J. De Leeuw and E. Meijer, editors, *Handbook of Multilevel Analysis*, chapter 1, pages 1–75. Springer Verlag, 2008.

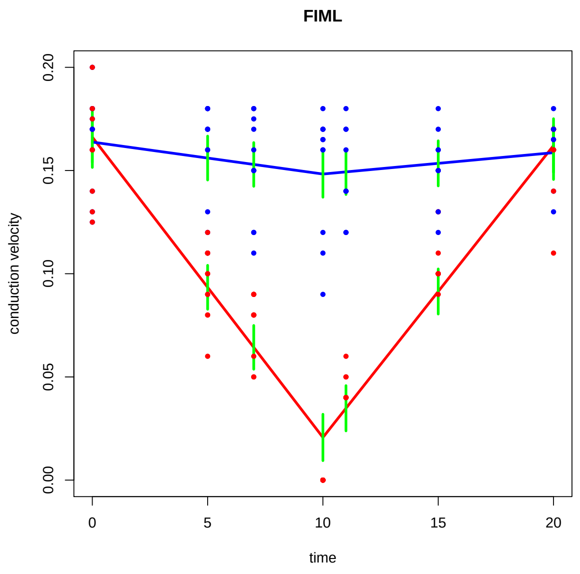

FIGURE 2.

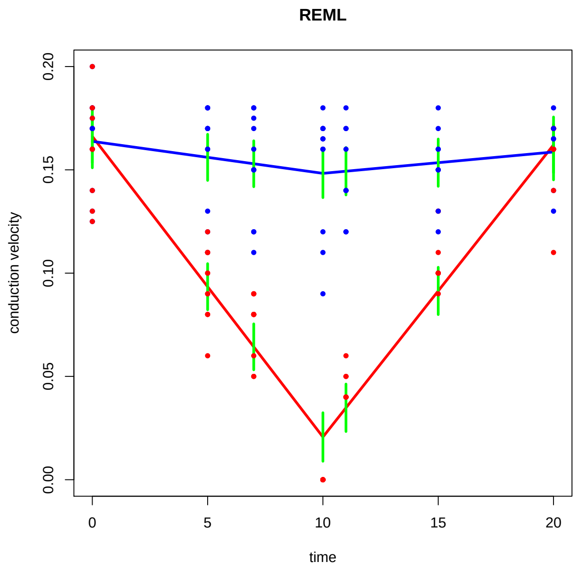

FIGURE 3.

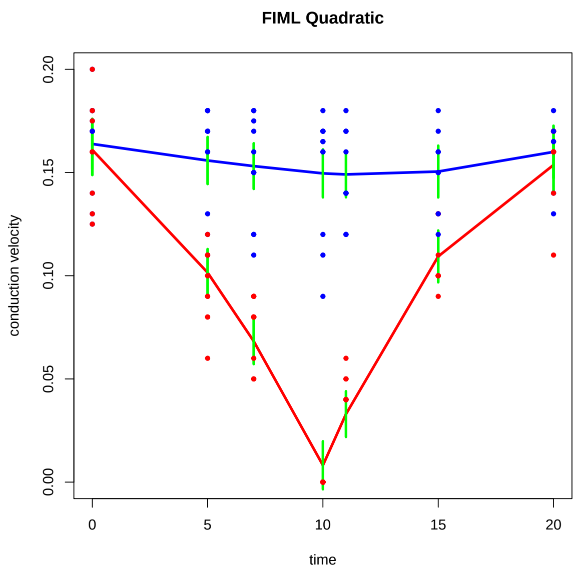

FIGURE 4.

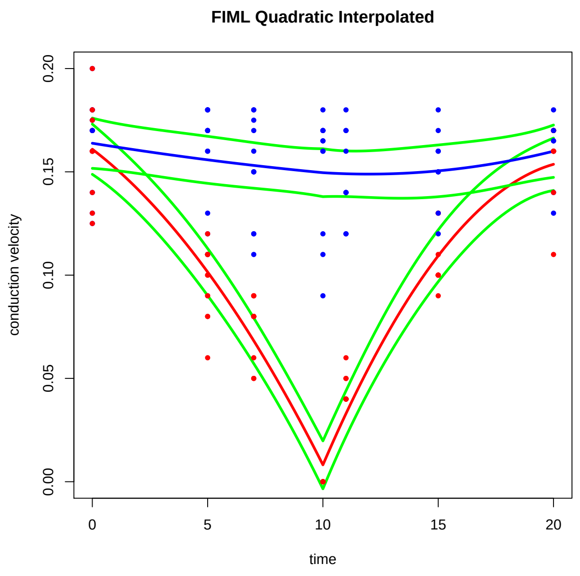

FIGURE 5.

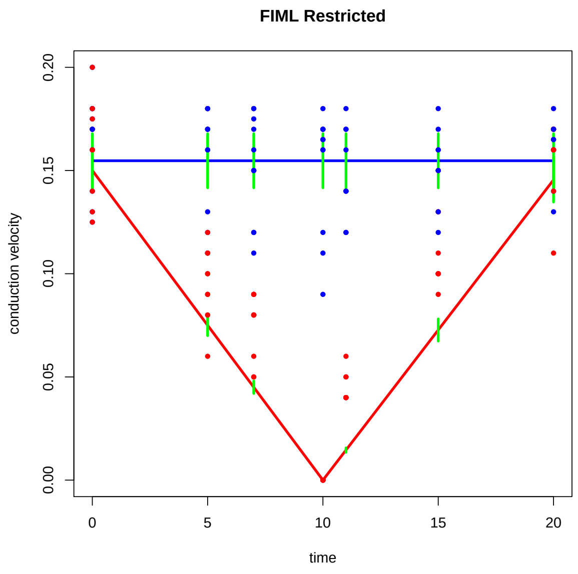

FIGURE 6.

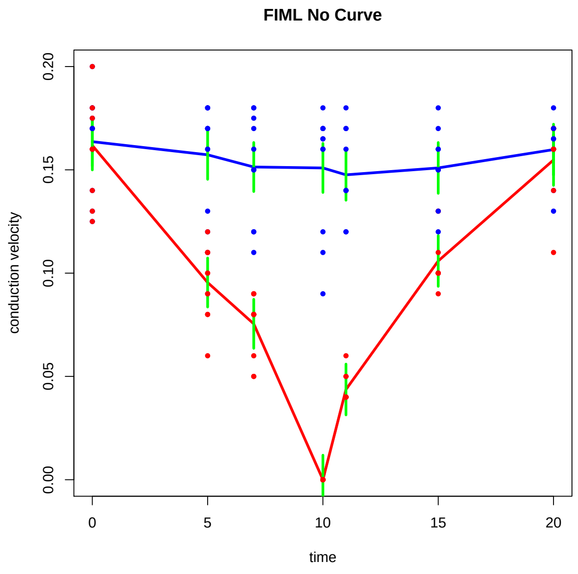

FIGURE 7.

APPENDIX A. CODE

A.1. Multilevel Package.

```
#
#   multilevel package
#   Copyright (C) 2008  Jan de Leeuw <deleeuw@stat.ucla.edu>
#   UCLA Department of Statistics, Box 951554, Los Angeles, CA 90095-1554
#
#   This program is free software; you can redistribute it and/or modify
#   it under the terms of the GNU General Public License as published by
#   the Free Software Foundation; either version 2 of the License, or
#   (at your option) any later version.
#
#   This program is distributed in the hope that it will be useful,
#   but WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#   GNU General Public License for more details.
#
#   You should have received a copy of the GNU General Public License
#   along with this program; if not, write to the Free Software
#   Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
###################################################################
#
# version 0.1, 2008-08-02   Initial  Release
# version 0.2, 2008-08-04   one-step and two-step initial estiates
# version 0.3, 2008-08-04   initial estimate xi
# version 1.0, 2008-08-04   added REML
# version 1.1, 2008-08-04   allow singularities in Z_j
# version 1.2, 2008-08-05   allow X_j to be of full rank
# version 2.0, 2008-08-06   rewrote everything using QR
# version 2.1, 2008-08-06   new initial VC estimate
# version 2.2, 2008-08-06   refactoring
# version 2.3, 2008-08-08   curve prediction
#

library(MASS)

multilevel<-function(allData,omStruc,full=TRUE,wlsini=1,itmax=100,epsg=1e-6,epsh=1
     e-6,verbose=TRUE){
ngrp<-length(allData); omPar<-length(omStruc); rgPar<-ncol(allData[[1]]$z); itel
     <-1
allMulti<-makeAllMulti(allData)
parStruc<-makeIniEst(allMulti,omStruc,wlsini)
repeat{
    llik<-gsg<-hsgsg<-0
    ggm<-rep(0,rgPar); gxi<-rep(0,omPar)
    hgmgm<-matrix(0,rgPar,rgPar); hsgxi<-rep(0,omPar)
    hxixi<-matrix(0,omPar,omPar)
    auxStruc<-list(llik=llik,ggm=ggm,gsg=gsg,gxi=gxi,
        hgmgm=hgmgm,hsgsg=hsgsg,hsgxi=hsgxi,hxixi=hxixi)
    auxStruc<-addFIML(auxStruc,parStruc,omStruc,allMulti)
    if (!full) auxStruc<-addREML(auxStruc,parStruc,omStruc,allMulti)
    llik<-auxStruc$llik
    ggm<-auxStruc$ggm; gsg<-auxStruc$gsg; gxi<-auxStruc$gxi
    hgmgm<-auxStruc$hgmgm; hsgsg<-auxStruc$hsgsg
    hsgxi<-auxStruc$hsgxi; hxixi<-auxStruc$hxixi
    hvarco<-rbind(c(hsgsg,hsgxi),cbind(hsgxi,hxixi))
    cvarco<-drop(ginv(hvarco)%*%c(gsg,gxi))
    cregre<-drop(ginv(hgmgm)%*%ggm)
    apsv<-max(abs(cvarco)); apsr<-max(abs(cregre))
    apsg<-max(abs(ggm)); apsh<-max(abs(c(gsg,gxi)))
    if (verbose) cat("Iteration: ",formatC(itel,width=3, format="d"),
        " NLLik: ", formatC(llik,digits=8,width=12,format="f"),
        " RChge: ",formatC(apsr,digits=8,width=12,format="f"),
        " VChge: ",formatC(apsv,digits=8,width=12,format="f"),
        " RGrad: ",formatC(apsg,digits=8,width=12,format="f"),
        " VGrad: ",formatC(apsg,digits=8,width=12,format="f"),
        "\n")
    if (((apsg<epsg) && (apsh<epsh)) || (itel == itmax)) break()
    parStruc$gm<-parStruc$gm-cregre
    parStruc$sg<-parStruc$sg-cvarco[1]
    parStruc$xi<-parStruc$xi-cvarco[-1]
    itel<-itel+1
    }
allPost<-makeAllPost(allMulti,parStruc,omStruc,auxStruc)
return(list(parStruc=parStruc,auxStruc=auxStruc,allPost=allPost,allMulti=allMulti)
     )
}

makeAllMulti<-function(allData)
{
ngrp<-length(allData)
oneMulti<-list(x=0,y=0,z=0,n=0,p=0,e=0,b=0,r=0,s=0)
allMulti<-repList(oneMulti,ngrp)
for (igrp in 1:ngrp) {
    x<-allData[[igrp]]$x
    y<-allData[[igrp]]$y
    z<-allData[[igrp]]$z
    ind<-which(!is.na(y))
    x<-x[ind,]; qq<-qr(x)
    k<-qq$rank; pv<-qq$pivot
    p<-allMulti[[igrp]]$p<-k
    x<-allMulti[[igrp]]$x<-qr.Q(qq)[,1:k]
    r<-allMulti[[igrp]]$r<-qr.R(qq)[1:k,order(pv)]
    n<-allMulti[[igrp]]$n<-length(ind)
    y<-allMulti[[igrp]]$y<-y[ind]
    allMulti[[igrp]]$z<-r%*%z
    b<-allMulti[[igrp]]$b<-colSums(y*x)
    e<-allMulti[[igrp]]$e<-y-drop(x%*%b)
    allMulti[[igrp]]$s<-if (n > p) (sum(e^2)/(n-p)) else 0
    }
return(allMulti)
}

```

```
makeAllPost<-function(allMulti,parStruc,omStruc,auxStruc) {
ngrp<-length(allMulti)
onePost<-list(bls=0,bml=0,bps=0,yls=0,yml=0,yps=0,vml=0)
allPost<-repList(onePost,ngrp)
for (igrp in 1:ngrp) {
    gm<-parStruc$gm
    xi<-parStruc$xi
    sg<-parStruc$sg
    r<-allMulti[[igrp]]$r
    p<-allMulti[[igrp]]$p
    om<-omMake(omStruc,r,xi)
    w<-om+sg*diag(p)
    z<-allMulti[[igrp]]$z
    x<-allMulti[[igrp]]$x
    v<-om%*%ginv(w)
    cvar<-ginv((auxStruc$hgmgm)/2)
    bml<-allPost[[igrp]]$bml<-drop(z%*%gm)
    bls<-allPost[[igrp]]$bls<-allMulti[[igrp]]$b
    bps<-allPost[[igrp]]$bps<-drop(bml+v%*%(bls-bml))
    allPost[[igrp]]$yls<-drop(x%*%bls)
    allPost[[igrp]]$yml<-drop(x%*%bml)
    allPost[[igrp]]$yps<-drop(x%*%bps)
    allPost[[igrp]]$vml<-sqrt(diag(tcrossprod(x%*%z,x%*%z%*%cvar)))
    }
return(allPost)
}

makeIniEst<-function(allMulti,omStruc,wlsini=1) {
ngrp<-length(allMulti); rgPar<-ncol(allMulti[[1]]$z)
omPar<-length(omStruc)
c<-matrix(0,rgPar,rgPar)
d<-rep(0,rgPar)
sp<-ss<-0
for (igrp in 1:ngrp) {
    z<-allMulti[[igrp]]$z
    x<-allMulti[[igrp]]$x
    b<-allMulti[[igrp]]$b
    n<-allMulti[[igrp]]$n
    p<-allMulti[[igrp]]$p
    s<-allMulti[[igrp]]$s
    r<-allMulti[[igrp]]$r
    sp<-sp+(n-p)
    ss<-ss+(n-p)*s
    if (wlsini == 0) {
        c<-c+crossprod(z)
        d<-d+drop(crossprod(z,b))
        }
    else {
        h<-x%*%z
        c<-c+crossprod(h)
        d<-d+drop(crossprod(h,x%*%b))
        }
    }
```

```
gm<-drop(ginv(c)%*%d)
if (sp > 0) sg<-ss/sp else sg<-0
g<-rep(0,omPar); c<-matrix(0,omPar,omPar)
f<-rep(0,omPar); d<-0; m<-0
for (igrp in 1:ngrp) {
    b<-allMulti[[igrp]]$b
    z<-allMulti[[igrp]]$z
    r<-allMulti[[igrp]]$r
    e<-b-drop(z%*%gm)
    h<-outer(e,e)
    for (imPar in 1:omPar) {
        tci<-r%*%omStruc[[imPar]]%*%t(r)
        g[imPar]<-g[imPar]+sum(h*tci)
        f[imPar]<-f[imPar]+sum(diag(tci))
        for (jmPar in 1:omPar)
            tcj<-r%*%omStruc[[jmPar]]%*%t(r)
            c[imPar,jmPar]<-c[imPar,jmPar]+sum(tci*tcj)
        }
    d<-d+sum(diag(h))
    m<-m+nrow(r)
    }
cc<-rbind(c(m,f),cbind(f,c))
dd<-c(d,g)
ee<-drop(ginv(cc)%*%dd)
return(list(gm=gm,sg=ee[1],xi=ee[-1]))
}

addFIML<-function(auxStruc,parStruc,omStruc,allMulti) {
ngrp<-length(allMulti); omPar<-length(omStruc)
llik<-auxStruc$llik
ggm<-auxStruc$ggm; gsg<-auxStruc$gsg; gxi<-auxStruc$gxi
hgmgm<-auxStruc$hgmgm; hsgsg<-auxStruc$hsgsg
hsgxi<-auxStruc$hsgxi; hxixi<-auxStruc$hxixi
for (igrp in 1:ngrp) {
    b<-allMulti[[igrp]]$b
    z<-allMulti[[igrp]]$z
    s<-allMulti[[igrp]]$s
    n<-allMulti[[igrp]]$n
    p<-allMulti[[igrp]]$p
    r<-allMulti[[igrp]]$r
    sg<-parStruc$sg
    gm<-parStruc$gm
    xi<-parStruc$xi
    om<-omMake(omStruc,r,xi)
    w<-om+sg*diag(p)
    u<-ginv(w)
    o<-u%*%u
    e<-b-drop(z%*%gm)
    v<-drop(u%*%e)
    t<-outer(v,v)-u
    llik<-llik+log(det(w))+sum(e*v)
    if (n > p) llik<-llik+(n-p)*(log(sg)+s/sg)
    ggm<-ggm-2*colSums(z*v)
    gsg<-gsg-sum(diag(t))
    if (n > p) gsg<-gsg-(n-p)*(s-sg)/(sg^2)
    hgmgm<-hgmgm+2*crossprod(z,u%*%z)
    hsgsg<-hsgsg+sum(diag(o))
    if (n > p) hsgsg<-hsgsg+((n-p)/(sg^2))
    for (imPar in 1:omPar) {
        omi<-r%*%omStruc[[imPar]]%*%t(r)
        hsgxi[imPar]<-hsgxi[imPar]+sum(omi*o)
        gxi[imPar]<-gxi[imPar]-sum(omi*t)
        for (jmPar in 1:omPar) {
            omj<-r%*%omStruc[[jmPar]]%*%t(r)
            hxixi[imPar,jmPar]<-hxixi[imPar,jmPar]+sum((u%*%omi%*%u)*omj)
            }
        }
    }
return(list(llik=llik,ggm=ggm,gsg=gsg,gxi=gxi,hgmgm=hgmgm,hsgsg=hsgsg,hsgxi=hsgxi,
     hxixi=hxixi))
}

addREML<-function(auxStruc,parStruc,omStruc,allMulti) {
ggm<-auxStruc$ggm
hgmgm<-auxStruc$hgmgm
a<-ginv(hgmgm/2)
omPar<-length(omStruc); ngrp<-length(allMulti); rgPar<-length(parStruc$gm)
llik<-auxStruc$llik
llik<-llik-log(det(a))
lbd<-matrix(0,rgPar,rgPar); psi<-repList(lbd,omPar)
for (igrp in 1:ngrp) {
    xi<-parStruc$xi
    sg<-parStruc$sg
    z<-allMulti[[igrp]]$z
    r<-allMulti[[igrp]]$r
    p<-allMulti[[igrp]]$p
    om<-omMake(omStruc,r,xi)
    w<-om+sg*diag(p)
    v<-solve(w,z)
    lbd<-lbd+crossprod(v)
    for (imPar in 1:omPar) {
        omi<-r%*%omStruc[[imPar]]%*%t(r)
        psi[[imPar]]<-psi[[imPar]]+crossprod(v,omi%*%v)
        }
    }
gsg<-auxStruc$gsg
gxi<-auxStruc$gxi
hsgsg<-auxStruc$hsgsg
hsgxi<-auxStruc$hsgxi
hxixi<-auxStruc$hxixi
gsg<-gsg-sum(a*lbd)
hsgsg<-hsgsg-sum(lbd*(a%*%lbd%*%a))
for (imPar in 1:omPar) {
    gxi[imPar]<-gxi[imPar]-sum(a*psi[[imPar]])
    hsgxi[imPar]<-hsgxi[imPar]-sum(lbd*(a%*%psi[[imPar]]*a))
    for (jmPar in 1:omPar)
```

```
        hxixi[imPar,jmPar]<-hxixi[imPar,jmPar]-sum(psi[[imPar]]*(a%*%psi[[jmPar]]
              *a))
    }
return(list(llik=llik,ggm=ggm,gsg=gsg,gxi=gxi,hgmgm=hgmgm,hsgsg=hsgsg,hsgxi=hsgxi,
     hxixi=hxixi))
}

omMake<-function(omStruc,r,xi) {
   omPar<-length(omStruc)
   omDim<-nrow(r)
   om<-matrix(0,omDim,omDim)
   for (imPar in 1:omPar) om<-om+xi[imPar]*(r%*%omStruc[[imPar]]%*%t(r))
   return(om)
}

omStrucRI<-function(n) {
omStruc<-list(matrix(0,n,n))
omStruc[[1]][n,n]<-1
return(omStruc)
}

omStrucDG<-function(n) {
omStruc<-repList(matrix(0,n,n),n)
for (i in 1:n)
    omStruc[[i]][i,i]<-1
return(omStruc)
}

makePredY<-function(xx,igrp,data,outStruc) {
z<-data[[igrp]]$z
cvar<-ginv((outStruc[[2]]$hgmgm)/2)
gm<-outStruc[[1]]$gm
xxml<-drop(xx%*%z%*%gm)
vxml<-sqrt(diag(tcrossprod(xx%*%z,xx%*%z%*%cvar)))
return(list(xxml,vxml))
}

repList<-function(x,n) {
z<-list()
for (i in 1:n)
    z<-c(z,list(x))
return(z)
}
```

A.2. **Data Creation.**

```
library(gdata)

mat1<-read.xls("./REPEATED ANALYSIS DATA BASE .xls",sheet=1)
mat1<-mat1[3:35,2:8]
mat1<-mat1[-(3*(1:11)-2),]
rownames(mat1)<-as.character(1:22)
```

```
mat1<-cbind(as.vector(t(matrix(1:11,11,2))),as.vector(matrix(c(0,1),11,2)),mat1)
names(mat1)<-c("sample","ischemia","0","5","7","10","11","15","20")
mat1<-apply(as.matrix(mat1),2,as.numeric)

mat2<-read.xls("./REPEATED ANALYSIS DATA BASE .xls",sheet=2)
mat2<-mat2[3:35,2:8]
mat2<-mat2[-(3*(1:11)-2),]
rownames(mat2)<-as.character(1:22)
mat2<-cbind(as.vector(t(matrix(1:11,11,2))),as.vector(matrix(c(0,1),11,2)),mat2)
names(mat2)<-c("sample","ischemia","0","5","7","10","11","15","20")
mat2<-apply(as.matrix(mat2),2,as.numeric)

mat3<-read.xls("./REPEATED ANALYSIS DATA BASE .xls",sheet=3)
mat3<-mat3[3:35,2:8]
mat3<-mat3[-(3*(1:11)-2),]
rownames(mat3)<-as.character(1:22)
mat3<-cbind(as.vector(t(matrix(1:11,11,2))),as.vector(matrix(c(0,1),11,2)),mat3)
names(mat3)<-c("sample","ischemia","0","5","7","10","11","15","20")
mat3<-apply(as.matrix(mat3),2,as.numeric)

mat4<-read.xls("./REPEATED ANALYSIS DATA BASE .xls",sheet=4)
mat4<-mat4[3:35,2:8]
mat4<-mat4[-(3*(1:11)-2),]
rownames(mat4)<-as.character(1:22)
mat4<-cbind(as.vector(t(matrix(1:11,11,2))),as.vector(matrix(c(0,1),11,2)),mat4)
names(mat4)<-c("sample","ischemia","0","5","7","10","11","15","20")
mat4<-apply(as.matrix(mat4),2,as.numeric)

`f` <-
structure(c(-10, -5, -3, 0, 0, 0, 0, 0, 0, 0, 0, 1, 5, 10, 1,
1, 1, 1, 1, 1, 1), .Dim = c(7L, 3L))

`f2` <-
structure(c(-10, -5, -3, 0, 0, 0, 0, 100, 25, 9, 0, 0, 0, 0,
0, 0, 0, 0, 1, 5, 10, 0, 0, 0, 0, 1, 25, 100, 1, 1, 1, 1, 1,
1, 1), .Dim = c(7L, 5L))

yy1<-matrix(0,99,3)
yy1[1:50,1]<-seq(0,10,length=50)-10
yy1[50:99,2]<-seq(10,20,length=50)-10
yy1[,3]<-1

yy2<-cbind(yy1[,1],yy1[,1]^2,yy1[,2],yy1[,2]^2,yy1[,3])

tm<-c(0,5,7,10,11,15,20)

xx<-seq(0,10,length=50)-10
zz<-seq(10,20,length=50)-10
yy<-matrix(0,99,3)
yy[1:49,1]<-xx[1:49]
yy[51:99,2]<-zz[2:50]
yy[,3]<-1

```

```
repList<-function(x,n) {
z<-list()
for (i in 1:n)
    z<-c(z,list(x))
return(z)
}

makeMLData<-function(mat){
ngrp<-nrow(mat)
oneData<-list(x=0,y=0,z=0)
allData<-repList(oneData,ngrp)
for (igrp in 1:ngrp) {
    allData[[igrp]]$x<-f
    allData[[igrp]]$y<-mat[igrp,3:9]
    allData[[igrp]]$z<-if (igrp%%2 == 1) cbind(diag(3),matrix(0,3,3)) else cbind(
         matrix(0,3,3),diag(3))
    }
return(allData)
}

makeML2Data<-function(mat){
ngrp<-nrow(mat)
oneData<-list(x=0,y=0,z=0)
allData<-repList(oneData,ngrp)
for (igrp in 1:ngrp) {
    allData[[igrp]]$x<-f2
    allData[[igrp]]$y<-mat[igrp,3:9]
    allData[[igrp]]$z<-if (igrp%%2 == 1) cbind(diag(5),matrix(0,5,5)) else cbind(
         matrix(0,5,5),diag(5))
    }
return(allData)
}

makeMLIData<-function(mat){
set.seed(12345)
x<-matrix(rnorm(49),7,7)
x[,1]<-1
x<-qr.Q(qr(x))[,rev(1:7)]
ngrp<-nrow(mat)
oneData<-list(x=0,y=0,z=0)
allData<-repList(oneData,ngrp)
for (igrp in 1:ngrp) {
    allData[[igrp]]$x<-x
    allData[[igrp]]$y<-mat[igrp,3:9]
    allData[[igrp]]$z<-if (igrp%%2 == 1) cbind(diag(7),matrix(0,7,7)) else cbind(
         matrix(0,7,7),diag(7))
    }
return(allData)
}


makeMLRData<-function(mat){
ngrp<-nrow(mat)
oneData<-list(x=0,y=0,z=0)
allData<-repList(oneData,ngrp)
for (igrp in 1:ngrp) {
    allData[[igrp]]$x<-f
    allData[[igrp]]$y<-mat[igrp,3:9]
    if (igrp%%2 == 1) {
        z<-cbind(diag(3),matrix(0,3,3))
        z[-3,]<-0
        }
    else {
        z<-cbind(matrix(0,3,3),diag(3))
        z[3,]<-0
        }
    allData[[igrp]]$z<-z
    }
return(allData)
}


data1<-makeMLData(mat1)
data2<-makeMLIData(mat2)
data3<-makeMLIData(mat3)
data4<-makeMLIData(mat4)
data5<-makeML2Data(mat2)
data6<-makeMLRData(mat2)
data7<-makeMLRData(mat1)
data8<-makeML2Data(mat1)
data9<-makeMLIData(mat1)
```

A.3. **Data Runs.**

```
outStruc1<-multilevel(data1,omStrucRI(3),full=TRUE,itmax=1000)
outPost1<-outStruc1[[3]]
pdf("cvolF.pdf")
yml<-outPost1[[22]]$yml; vml<-outPost1[[22]]$vml
plot(tm,yml,type="l",ylim=c(0,.20),xlab="time",ylab="conduction velocity",col="RED
     ",lwd=3,main="FIML")
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
yml<-outPost1[[21]]$yml; vml<-outPost1[[21]]$vml
lines(tm,yml,col="BLUE",lwd=3)
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
for (i in 1:22) if (i%%2 == 0) points(tm,mat1[i,3:9],col="RED",pch=20) else
     points(tm,mat1[i,3:9],col="BLUE",pch=20)
dev.off()

outStruc7<-multilevel(data7,omStrucRI(3),full=TRUE,itmax=1000)
outPost7<-outStruc7[[3]]
pdf("cvolRR.pdf")
yml<-outPost7[[22]]$yml; vml<-outPost7[[22]]$vml
```

```
plot(tm,yml,type="l",ylim=c(0,.20),xlab="time",ylab="conduction velocity",col="RED
     ",lwd=3,main="FIML Restricted")
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
yml<-outPost7[[21]]$yml; vml<-outPost7[[21]]$vml
lines(tm,yml,col="BLUE",lwd=3)
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
for (i in 1:22) if (i%%2 == 0) points(tm,mat1[i,3:9],col="RED",pch=20) else
     points(tm,mat1[i,3:9],col="BLUE",pch=20)
dev.off()

outStruc8<-multilevel(data8,omStrucRI(5),full=TRUE,itmax=1000)
outPost8<-outStruc8[[3]]
pdf("cvolR2.pdf")
yml<-outPost8[[22]]$yml; vml<-outPost8[[22]]$vml
plot(tm,yml,type="l",ylim=c(0,.20),xlab="time",ylab="conduction velocity",col="RED
     ",lwd=3,main="FIML Quadratic")
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
yml<-outPost8[[21]]$yml; vml<-outPost8[[21]]$vml
lines(tm,yml,col="BLUE",lwd=3)
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
for (i in 1:22) if (i%%2 == 0) points(tm,mat1[i,3:9],col="RED",pch=20) else
     points(tm,mat1[i,3:9],col="BLUE",pch=20)
dev.off()

outStruc9<-multilevel(data9,omStrucRI(7),full=TRUE,itmax=1000)
outPost9<-outStruc9[[3]]
pdf("cvolU.pdf")
yml<-outPost9[[22]]$yml; vml<-outPost9[[22]]$vml
plot(tm,yml,type="l",ylim=c(0,.20),xlab="time",ylab="conduction velocity",col="RED
     ",lwd=3,main="FIML No Curve")
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
yml<-outPost9[[21]]$yml; vml<-outPost9[[21]]$vml
lines(tm,yml,col="BLUE",lwd=3)
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
for (i in 1:22) if (i%%2 == 0) points(tm,mat1[i,3:9],col="RED",pch=20) else
     points(tm,mat1[i,3:9],col="BLUE",pch=20)
dev.off()

outStruc2<-multilevel(data2,omStrucRI(7),full=TRUE,itmax=1000)
outPost2<-outStruc2[[3]]
pdf("APaF.pdf")
yml<-outPost2[[2]]$yml; vml<-outPost2[[2]]$vml
plot(tm,yml,type="l",ylim=c(0,1.20),xlab="time",ylab="AP Amplitude",col="RED",lwd
     =3,main="FIML No Curve")
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
yml<-outPost2[[1]]$yml; vml<-outPost2[[1]]$vml
```

```
lines(tm,yml,col="BLUE",lwd=3)
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
for (i in 1:22) if (i%%2 == 0) points(tm,mat2[i,3:9],col="RED",pch=20) else
     points(tm,mat2[i,3:9],col="BLUE",pch=20)
dev.off()

outStruc3<-multilevel(data3,omStrucRI(7),full=TRUE,itmax=1000)
outPost3<-outStruc3[[3]]
pdf("APDF.pdf")
yml<-outPost3[[22]]$yml; vml<-outPost3[[22]]$vml
plot(tm,yml,type="l",ylim=c(0,110),xlab="time",ylab="APD",col="RED",lwd=3,main="
     FIML No Curve")
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
yml<-outPost3[[21]]$yml; vml<-outPost3[[21]]$vml
lines(tm,yml,col="BLUE",lwd=3)
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
for (i in 1:22) if (i%%2 == 0) points(tm,mat3[i,3:9],col="RED",pch=20) else
     points(tm,mat3[i,3:9],col="BLUE",pch=20)
dev.off()

outStruc4<-multilevel(data4,omStrucRI(7),full=TRUE,itmax=1000)
outPost4<-outStruc4[[3]]
pdf("CatF.pdf")
yml<-outPost4[[2]]$yml; vml<-outPost4[[2]]$vml
plot(tm,yml,type="l",ylim=c(0,200),xlab="time",ylab="Ca transient",col="RED",lwd
     =3,main="FIML No Curve")
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
yml<-outPost4[[1]]$yml; vml<-outPost4[[1]]$vml
lines(tm,yml,col="BLUE",lwd=3)
for (i in 1:7) lines(c(tm[i],tm[i]),c(yml[i]-2*vml[i],yml[i]+2*vml[i]),col="GREEN"
     ,lwd=3)
for (i in 1:22) if (i%%2 == 0) points(tm,mat4[i,3:9],col="RED",pch=20) else
     points(tm,mat4[i,3:9],col="BLUE",pch=20)
dev.off()
```

DEPARTMENT OF STATISTICS, UNIVERSITY OF CALIFORNIA, LOS ANGELES, CA 90095-1554

E-mail address, Jan de Leeuw: `deleeuw@stat.ucla.edu`

URL, Jan de Leeuw: `http://gifi.stat.ucla.edu`