Practical 4: Linear Mixed Models

A *linear mixed model* has the general form

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon}, \text{ where } \mathbf{b} \sim N(\mathbf{0}, \boldsymbol{\psi}_\theta) \text{ and } \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I}\sigma^2).$$

It differs from a linear model by having the extra term $\mathbf{Zb}$ where $\mathbf{Z}$ is an $n \times p$ model matrix associated with the *random effects* $\mathbf{b}$. The latter are Gaussian random variables with covariance matrix $\boldsymbol{\psi}_\theta$, which usually has some simple structure and is parameterized by a small number of parameters in vector $\boldsymbol{\theta}$. The idea is to allow the random component of a linear model the sort of rich structure allowed for the mean. The models are useful when randomness is present at different levels within the data (e.g. each patient has a randomly different blood-pressure effect, while each blood pressure measurement on an individual patient has a random measurement error).

In this practical we will assume that $\boldsymbol{\psi}_\theta$ is diagonal. That is that the random effects are independent. We will also assume that several elements of $\mathbf{b}$ have a common variance, determined by a single element of $\boldsymbol{\theta}$. Exactly how to set this up will be described below. From basic probability you can see that $E(\mathbf{y}) = \mathbf{X}\boldsymbol{\beta}$ and that the covariance matrix for $\mathbf{y}$ is $\mathbf{Z}\boldsymbol{\psi}_\theta\mathbf{Z}^T + \mathbf{I}\sigma^2$. Hence $\mathbf{y} \sim N(\mathbf{X}\boldsymbol{\beta}, \mathbf{Z}\boldsymbol{\psi}_\theta\mathbf{Z}^T + \mathbf{I}\sigma^2)$, so that (dropping uninteresting constants) the log likelihood for $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ is

$$-(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Z}\boldsymbol{\psi}_\theta\mathbf{Z}^T + \mathbf{I}\sigma^2)^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})/2 - \log|\mathbf{Z}\boldsymbol{\psi}_\theta\mathbf{Z}^T + \mathbf{I}\sigma^2|/2$$

This expression is rather inefficient to work with if n is substantially larger than p, because of the inverse of an $n \times n$ matrix in the first term. A computationally efficient approach is based on forming the QR decomposition

$$\mathbf{Z} = \mathbf{Q}\begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}.$$

Replacing $\mathbf{Z}$ by its QR decomposition we then have

$$\mathbf{Z}\boldsymbol{\psi}_\theta\mathbf{Z}^T + \mathbf{I}\sigma^2 = \mathbf{Q}\begin{bmatrix} (\mathbf{R}\boldsymbol{\psi}_\theta\mathbf{R}^T + \mathbf{I}_p\sigma^2) & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{n-p}\sigma^2 \end{bmatrix}\mathbf{Q}^T$$

and, by definition of $\mathbf{W}$

$$\mathbf{W} = (\mathbf{Z}\boldsymbol{\psi}_\theta\mathbf{Z}^T + \mathbf{I}\sigma^2)^{-1} = \mathbf{Q}\begin{bmatrix} (\mathbf{R}\boldsymbol{\psi}_\theta\mathbf{R}^T + \mathbf{I}_p\sigma^2)^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{n-p}\sigma^2 \end{bmatrix}\mathbf{Q}^T. \quad (1)$$

The point here is that multiplication by $\mathbf{Q}$/$\mathbf{Q}^T$ is efficient using `qr.qy`/`qr.qry`, while the RHS of (1) involves only a $p \times p$ matrix inversion. It also follows that

$$\log|\mathbf{Z}\boldsymbol{\psi}_\theta\mathbf{Z}^T + \mathbf{I}\sigma^2| = \log|\mathbf{R}\boldsymbol{\psi}_\theta\mathbf{R}^T + \mathbf{I}_p\sigma^2| + (n-p)\log(\sigma^2)$$

Cholesky decomposition should be used to find the first term on the RHS above, and will also enable computations involving the matrix inverse when multiplying by $\mathbf{W}$. Note that given a value of $\boldsymbol{\theta}$ the MLE of $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{y}. \quad (2)$$

Given $\mathbf{X}^T\mathbf{W}\mathbf{X}$ and $\mathbf{X}^T\mathbf{W}\mathbf{y}$ this can efficiently be computed with the help of another Cholesky decomposition. When computing $\mathbf{W}\mathbf{y}$ and $\mathbf{W}\mathbf{X}$ it is important to use the RHS of (1) to do this efficiently (and never to form $\mathbf{W}$ explicitly).

When estimating the LMM, a general purpose optimizer is used to find the MLE of $\boldsymbol{\theta}$, with the $\hat{\boldsymbol{\beta}}$ corresponding to any trial $\boldsymbol{\theta}$ being computed directly by (2) as part of evaluating the log likelihood (this sort of approach – where MLEs for some parameters can be computed directly – is sometimes known as *profile likelihood*).

Your task is to write a function for simple linear mixed model estimation

```
lmm(form,dat,ref=list())
```

- `form` is a model formula (as used in `lm`) and will specify the $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ part of the model. The variables used in the formula must be supplied in data frame `dat`.
- `dat` is the data frame containing all the variables needed in the model.

- `ref` is a list of vectors of variable names specifying the random effects for the $\mathbf{Zb}$ part of the model. There is a group of columns of $\mathbf{Z}$ for each vector. The columns are the model matrix for the interaction of the corresponding variables. For example if the jth vector in `ref` is `c("z","x","w3")` then the jth block of columns of $\mathbf{Z}$ will be given by `model.matrix(~ z:x:w3-1,dat)`. For a vector in `ref` containing a single variable name the block of columns are just those corresponding to the single variable; e.g. if the vector is `"foo"` then the model matrix columns are given by `model.matrix(~ foo-1,dat)`. The elements of $\mathbf{b}$ corresponding to each of these blocks of columns are independent with the same variance (so there will be one variance parameter for each block). If `ref` is empty then there are no random effects (your function should still work). `as.formula` may be useful.

The function should estimate the model parameters, $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ where the first element of θ is $\log \sigma$ and the remaining elements are the log standard deviations of the random effect variances. The function should return a list containing the MLEs in `beta` and `theta`.

1. You probably want to structure your code into at least 3 functions `lmm`; `LMMsetup`, to set up $\mathbf{Z}$, $\mathbf{X}$ and other things that need to be setup only once; `LMMprof`, to evaluate the (negative) log likelihood for a given $\boldsymbol{\theta}$, and compute the corresponding $\hat{\boldsymbol{\beta}}$ (hint: attributes may be useful for returning $\hat{\boldsymbol{\beta}}$).
2. You can use `optim` within `lmm` to optimize `LMMprof` w.r.t. $\boldsymbol{\theta}$.
3. You should compute efficiently as described above. So `solve`, `chol2inv`, `eigen`, `qr.Q`, `determinant` or `det` should not be used, and you should avoid unnecessary repetition of expensive operations.
4. Models of the sort fitted by `lmm` can also be estimated by the `lme4` package. One test of your code is to use the `Machines` data frame from package `nlme` and compare

   ```
   library(nlme);library(lme4)
   lmm(score ~ Machine,Machines,list("Worker",c("Worker","Machine")))
   lmer(score ~ Machine + (1|Worker) + (1|Worker:Machine),data=Machines,
         REML=FALSE)
   ```

   which should give the same parameter estimates. Do use the `debug` package.
5. You should also test your code on other examples, including the case of no random effects.
6. Do not include your tests in what you submit. Submit only functions in a single text file (.r).

Code should be submitted on Learn by 12 noon Friday 15th November 2024.

Marking Scheme: Full marks will be obtained for code that:

1. does what it is supposed to do efficiently, has been coded in R approximately as indicated, without requiring packages outwith what comes with base R, and follows the specification for `lmm`'s inputs and outputs exactly.
2. is carefully commented, so that someone reviewing the code can easily tell exactly what it is for, what it is doing and how it is doing it without having read this sheet, or knowing anything else about the code. Note that *easily tell* implies that the comments must also be as clear and *concise* as possible. You should assume that the reader knows basic R, but not that they know exactly what every function in R does.
3. is well structured and laid out, so that the code itself, and its underlying logic, are easy to follow.
4. was prepared collaboratively using git and github in a group of 3.
5. contains no evidence of having been copied, in whole or in part, from anyone else (AI counts as anyone else for this purpose), other students on this course, students at other universities (there are now tools to help detect this, including internationally), online sources, ChatGPT etc.
6. includes a comment stating team member contributions.

Individual marks may be adjusted within groups if contributions are widely different.