The lmekin function

Terry Therneau
Mayo Clinic

December 28, 2011

1 Background

The original kinship library had an implementation of linear mixed effects models using the matrix code found in coxme. Since the primary motivation for the functions in that library was to fit models with random family effects, i.e., using a kinship matrix for the correlation, the name *lmekin* was chosen. The reason for the program was entirely to check our arithmetic: the result of the matrix manipulations contained in it should give exactly the same answer as lme, and since the underlying routines were shared with coxme that gave a validity check for parts of coxme. With more time and a larger test suite the routine is no longer necessary for this purpose, however, it became popular with users (they often do unanticipated things) since it can fit a few models that lme cannot. Let me emphasis this: most models that can be fit with the lmekin function can also be fit with lme and/or lmer. For any such model the lme/lmer functions will be faster and have superior support routines (residuals, printing, plotting, etc.) The solution code for lmer is likely also more reliable since it has been exercised on a much wider variety of data sets.

However, there are models that lmekin will fit which lme will not. The most obvious of these are models with a random genetic effect, e.g. a kinship matrix. The second class will be models for which the user has written their own variance extension, as described in the variance vignette.

The follow-up methods for lmekin are limited, which reflects the fact that linear mixed effects models are not a primary focus for me, the author of the coxme package. A primary reason to update lmekin at all is a desire to depreciate the original kinship package; this routine was the last bit of functionality that is not otherwise available. The set of models fit by lmekin was also extended to include all of the random effects structures supported by coxme, which should make the routine more valuable. Contributions by others with deeper interest will be warmly received. Nevertheless, the core code is solid and reliable to the best of my ability and will be actively maintained.

2 Simple Models

The control code for lmekin is identical to coxme with respect to specifying the random effects, and both are modeled on the methods used in lmer. Here is a simple example using one of the data sets from Pinheiro and Bates.

```
> library(coxme)
> fit1 <- lme(effort~Type, random= ~ 1|Subject,data=ergoStool,
              method="ML")
> fit2 <- lmekin(effort ~ Type + (1|Subject), data=ergoStool,
                             method="ML")
> print(fit1)

Linear mixed-effects model fit by maximum likelihood
  Data: ergoStool
  Log-likelihood: -61.07222
  Fixed: effort ~ Type
(Intercept)      TypeT2      TypeT3      TypeT4
  8.5555556   3.8888889   2.2222222   0.6666667

Random effects:
 Formula: ~1 | Subject
        (Intercept) Residual
StdDev:     1.25626 1.037368

Number of Observations: 36
Number of Groups: 9

> print(fit2)

Linear mixed-effects kinship model fit by maximum likelihood
  Data: ergoStool
  Log-likelihood = -61.07222
  n= 36


Model:  effort ~ Type + (1 | Subject)
Fixed coefficients
                Value Std Error     z       p
(Intercept) 8.5555556 0.5430696 15.75 0.0e+00
TypeT2      3.8888889 0.4890198  7.95 1.8e-15
TypeT3      2.2222222 0.4890198  4.54 5.5e-06
TypeT4      0.6666667 0.4890198  1.36 1.7e-01

Random effects
 Group   Variable  Std Dev  Variance
 Subject Intercept 1.256260 1.578189
Residual error= 1.037368
```

And here is a slightly more complex one based on data from J. Cortinas [2]. There are 37 centers of varying size, and the simulated data set has both random intercepts and treatment effects per center.

```
> tdata <-eortc
> tdata$center2 <- factor(tdata$center)
> fit3 <- lme(y ~ trt, random= ~ trt|center2, data=tdata,
             method="ML")
> fit3

Linear mixed-effects model fit by maximum likelihood
  Data: tdata
  Log-likelihood: -19413.23
  Fixed: y ~ trt
(Intercept)         trt
  2200.3256   -571.2248

Random effects:
 Formula: ~trt | center2
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev    Corr
(Intercept)  146.0512 (Intr)
trt          227.1224 0.254
Residual    1017.2737

Number of Observations: 2323
Number of Groups: 37

> fit4 <- lmekin(y ~ trt + (1+ trt|center), tdata)
> fit4

Linear mixed-effects kinship model fit by maximum likelihood
  Data: tdata
  Log-likelihood = -19413.23
  n= 2323


Model:  y ~ trt + (1 + trt | center)
Fixed coefficients
                Value Std Error     z p
(Intercept) 2200.3482  47.60675 46.22 0
trt         -571.2595  61.90075 -9.23 0

Random effects
 Group  Variable  Std Dev    Variance   Corr
 center Intercept   141.2490 19951.2793   326.9359
        trt         230.2763 53027.1517
Residual error= 1017.271

> all.equal(fit3$logLik, fit4$loglik)
```

```
[1] TRUE
```

First note that the two fits give identical log-likelihoods, even though the coefficients differ. The log-likelihood function is somewhat flat on top, and because of different default starting estimates the two programs do not end up at exactly the same place.

One small difference above is that lmekin is a little more forgiving with respect to groups. The center variable in the eortc data set is numeric, when it appears on the right hand side of the vertical bar `(1 + trt|center)` the program assumes it is a grouping effect. The lme routine insists that the grouping variable be a factor. (In defense of lme, if one were to accidentally put a continuous variable on the right such as age, which has no business being there, the error message is welcome.)

A more important difference from lme (and lmer) is the inclusion of random intercepts. In lmer a random term like `(age | group` will actually fit the model `(1+age | group)`, i.e., an intercept term is assumed unless it is specifically removed by adding `-1` to the model. In lmekin an intercept is not assumed, the random effect you type is the one that you get. The primary reason for this is that lmer mimics lm, which also adds an intercept unless it is explicitly suppressed. The coxme function mimics coxph, which does not add an intercept. Since lmekin is built on the same routines as coxme it also follows that convention. (In Cox models there is not an intercept term for the fixed effects since this is absorbed into the baseline hazard).

3 Computation

The random effects linear model is

$$y = X\beta + Zb + \epsilon \tag{1}$$

$$b \sim N(0, \sigma^2 A(\theta) \tag{2}$$

$$\epsilon = N(0, \sigma^2) \tag{3}$$

Here β are the fixed and b the random coefficients, and the variance matrix A of the random effects depends on some arbitrary vector of parameters θ. For any fixed value of θ the solution for the remaining parameters is based on a QR decomposition, exactly as is laid out in section 2.2 of Pinheiro and Bates ([1]), leading also a profile likelihood value $L(\theta)$.

Notice that both β and σ can be solved for explicitly when θ is known, and that the variance of b is $\sigma^2 A$ not A. Thus for iteration, A will contain relative variances for components of b, something that Pinheiro and Bates refer to as the *precision* matrix. (See section 2.1.1; they also use a Cholesky decomposition Δ of the inverse precision for further numerical accuracy. We do not do so.) When the results of a fit are printed out these two are multiplied to give the variance of b directly. This decomposition will be invisible to most users, unless they either set initial values or retrieve variances directly from the coxme object.

The Z matrix is often sparse, so the QR computations are done using the Matrix library to take advantage of this. Maximization of $L(\theta)$ with respect to θ is accomplished with the optim() function.

References

[1] José C. Pinheiro and Douglas M. Bates, *Mixed-Effects Models in S and S-PLUS*, Springer, 2000.

[2] Cortinas Abrahantes, Jose; Burzykowski, Tomasz, A version of the EM algorithm for proportional hazards models with random effects, *Lecture Notes of the ICB Seminars*, p. 15-20, 2002.