The lmekin function

Terry Therneau
Mayo Clinic

May 11, 2018

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)
> require(nlme)
> 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 0.5430715 15.75 0.0e+00
TypeT2      3.8888889 0.4890188  7.95 1.8e-15
TypeT3      2.2222222 0.4890188  4.54 5.5e-06
TypeT4      0.6666667 0.4890188  1.36 1.7e-01

Random effects
 Group   Variable  Std Dev  Variance
 Subject Intercept 1.256269 1.578213
Residual error= 1.037366
```

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) 2200.3222  47.60846 46.22 0
trt         -571.2218  61.88263 -9.23 0

Random effects
 Group  Variable  Std Dev      Variance     Corr
 center Intercept 1.460461e+02 2.132947e+04 2.676947e+08
        trt       2.271210e+02 5.158395e+04
Residual error= 1017.273

> 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 GAW example

The following examples use data from one of the Genetic Analaysis Workshops (I don't remeber which year). First read in the saved data, create the pedigrees, and create the kinship matrix. Figure 1 shows a plot of the smallest of the 23 families in the file.

```
> require(kinship2)
> load("gaw.rda")
> gped <- with(gdata, pedigree(id, father, mother, sex=sex, famid=famid))
> kmat <- kinship(gped)
> plot(gped[9])
> gfit0 <- lm(age ~ q1, gdata)
> summary(gfit0)

Call:
lm(formula = age ~ q1, data = gdata)

Residuals:
   Min     1Q Median     3Q    Max
-42.67 -11.74   0.80  10.89  35.28

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.9795     2.3258   2.571   0.0103
q1            2.2059     0.1299  16.984   <2e-16

Residual standard error: 15.05 on 998 degrees of freedom
  (497 observations deleted due to missingness)
```

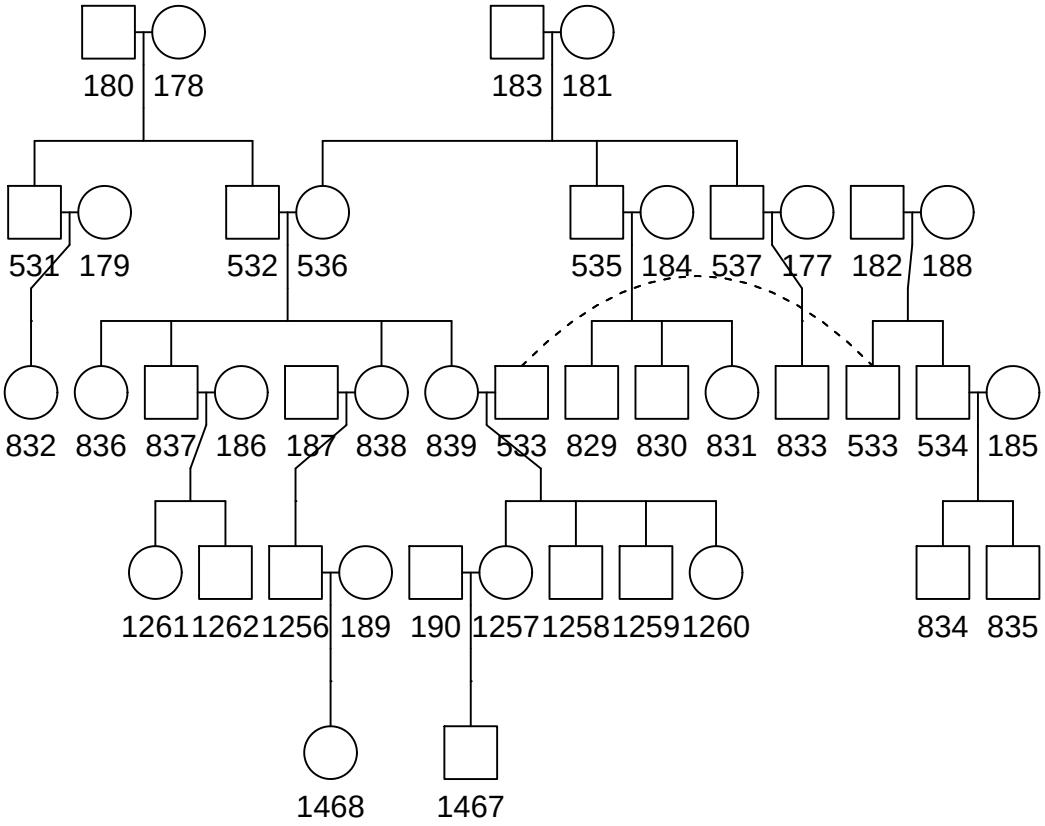

Figure 1: Pedigree 9 from the GAW data.

```
Multiple R-squared:  0.2242,        Adjusted R-squared:  0.2235
F-statistic: 288.5 on 1 and 998 DF,  p-value: < 2.2e-16

> gfit1 <- lmekin(age ~ q1 + (1|id), data=gdata, varlist=kmat*2)
> gfit1

Linear mixed-effects kinship model fit by maximum likelihood
  Data: gdata
  Log-likelihood = -4114.317
  n=1000 (497 observations deleted due to missingness)

Model:  age ~ q1 + (1 | id)
Fixed coefficients
               Value Std Error     z    p
(Intercept) 3.896828 2.4235529  1.61 0.11
q1          2.393351 0.1320173 18.13 0.00

Random effects
 Group Variable Std Dev   Variance
 id    Vmat.1    7.489286 56.089399
Residual error= 12.99801
```

The fit predicts age at onset using one quantitative trait along with a familial affect. The residual error is decreased when we include a familial effect, and the familial effects is substantial. The kinship matrix has diagonal elements of .5 (if there is no inbreeding); it is traditional to use a scaled version with elements of 1 in genetics models.

A next step is to look at the effect of a particular locus. The saved rda file also contains the results of a single SOLAR run at locus 6.90 along with the `pedindex` file created by SOLAR. We need to convert these into sparse matrix form, and add appropriate labels. (When there are kinship or ibd matrices, the coxme routine uses the matrix labels to match the proper row/col to the proper subject). The SOLAR package may reorder subjects in the data set; the pedindex matrix contains the new subject and family numbers in colums 1 and 6, and the original family and subject values in the last two columns. In this data set each subject has a unique identifier, so we do not need to include the family id in the matrix dimnames to obtain correct matches.

```
> sid <- pedindex[,9]
> ibd6.90 <- with(solar6.90, sparseMatrix(id2, id1, x=x, symmetric=TRUE,
                                          dimnames=list(sid, sid)))
> gfit2 <- lmekin(age ~ (1|id), data=gdata,
                  varlist= list(kmat, ibd6.90))
> print(gfit2)

Linear mixed-effects kinship model fit by maximum likelihood
  Data: gdata
  Log-likelihood = -4252.44
  n=1000 (497 observations deleted due to missingness)
```

```
Model:  age ~ (1 | id)
Fixed coefficients
               Value Std Error     z p
(Intercept) 45.59829  0.638492 71.42 0

Random effects
 Group Variable Std Dev   Variance
 id    Vmat.1    5.850013 34.222655
       Vmat.2    4.362513 19.031515
Residual error= 15.98072
```

The specific effect is modest for this locus: it partitions the familial effect found above into about 1/3 locus specific and 2/3 multifactorial. Another possible fit is to assume a common environmental effect for each family. (For pedigrees this large I have serious doubts about the relevance of the model below, but it serves as an illustration). When there are multiple random terms the varlist argument is matched up to them one by one, with the default choice used for any remaining, so in the model below the first will be a kinship effect and the second an uncorrelated random intercept per family.

```
> gfit3 <- lmekin(age ~ q1 + (1|id) + (1|famid), data=gdata,
                  varlist=kmat)
> gfit3

Linear mixed-effects kinship model fit by maximum likelihood
  Data: gdata
  Log-likelihood = -4114.079
  n=1000 (497 observations deleted due to missingness)

Model:  age ~ q1 + (1 | id) + (1 | famid)
Fixed coefficients
               Value Std Error     z    p
(Intercept) 3.861551 2.4377722  1.58 0.11
q1          2.394771 0.1319504 18.15 0.00

Random effects
 Group Variable  Std Dev    Variance
 id    Vmat.1     10.284711 105.775283
 famid Intercept   1.371693   1.881541
Residual error= 13.059
```

If one wanted to be specific the above model could be written as below, to identify the actual variance functions used for each.

```
> lmekin(age ~ q1 + (1|id) + (1|famid), data=gdata,
        varlist=list(coxmeMlist(kmat), coxmeFull))
```

4 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 $L(\theta)$.

For known A, this is solved as an augmented least squares problem with

$$y^* = \begin{pmatrix} y \\ 0 \end{pmatrix} \qquad X^* = \begin{pmatrix} X \\ 0 \end{pmatrix} \qquad Z^* = \begin{pmatrix} Z \\ \Delta \end{pmatrix}$$

where $\Delta'\Delta = A^{-1}$. The dummy rows of data have $y = 0$, $X = 0$ and Δ as the predictor variables. With known Δ, this gives the solution to all the other parameters as an ordinary least squares problem, which is solved using a QR decompostion. 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.

Thus, during the solution process A will contain relative variances for components of b, something that Pinheiro and Bates refer to as the *precision* matrix. When the results of a fit are printed out A is multiplied by σ^2 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. Initial values are for the parameters θ of A, and the results of the VarCorr function will also be terms of θ, not multiplied by the residual variance. This causes a complication if a user wanted to fix the the overall variance of the random effect at some constant; no solution to this is yet in place. For comparison see section 2.1.1 of Pinheiro and Bates. They also use the values of the Cholesky decomposition Δ directly as the unknowns for the optim function. This has the advantage of further numerical precision, avoids computing the Cholesky decompostion anew at each iteration, and guarrantees that the variance matrix is positive definite. However, though it works well for an unstructured variance, the lme default, the common genetic models do not have a simple representation in the Cholesky space and so we work directly with A.

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.