The `lme` function

Peter Dalgaard

Department of Biostatistics
University of Copenhagen

Mixed Models in R, January 2006
Introduction

- lme is the predecessor of lmer
- It has a more complicated syntax, and is not quite as fast
- But it is also more stable
- ... and will fit some models that lmer can not fit
Overview

- Basic model
- Some simple examples
- Grouped data classes
- Rudimentary Lattice (Trellis) graphics
- Specifying random components
Model

Laird and Ware (1982):

\[ y_i = X_i \beta + Z_i b_i + \epsilon_i \quad i = 1, \ldots, M \]
\[ b_i \sim N(0, \Psi), \quad \epsilon_i \sim N(0, \sigma^2 I) \]

- \( \beta \) fixed effects (p-vector)
- \( b_i \) random effects (q-vectors)
- \( X_i \) matrix \((n_i \times p)\)
- \( Z_i \) matrix \((n_i \times q)\)
Multilevel extension

Pinheiro and Bates use the following notation

\[
\begin{align*}
y_{ij} &= X_{ij} \beta + Z_{i,j} b_i + Z_{ij} b_{ij} + \epsilon_{ij} \\
i &= 1, \ldots, M \quad j &= 1, \ldots, M_i \\
b_i &\sim N(0, \Psi_1), \quad b_{ij} \sim N(0, \Psi_2), \quad \epsilon_i \sim N(0, \sigma^2 I)
\end{align*}
\]

- \(b_i\) level-1 random effects (\(q_1\)-vectors)
- \(b_{ij}\) level-2 random effects (\(q_2\)-vectors)
- \(Z_{i,j}\) matrix (\(n_{ij} \times q_1\)) (Notice the comma!)
- \(Z_{ij}\) matrix (\(n_{ij} \times q_2\))
- Extends to more than two levels
Basic usage

```r
library(nlme)
data(Rail)
lset(col.whitebg())
plot(Rail)
lme(travel~1, data=Rail, random=~1|Rail)
```
Remarks

- Standard one-way ANOVA with random variation
- Variation between rails larger than expected from variation within same rail
- Notice specification \texttt{random=\sim 1 | Rail}
- The DF calculation is not quite right.
Grouped Data objects

- The **Rail** object is a generalized data frame encoding the grouping of data
- That’s why it works just to say `plot(Rail)`
- Default formulas can use this information, `lme(travel 1, data=Rail)` or even `lme(Rail)` would suffice
Plotting residuals

r1 <- lme(travel~1, data=Rail)
plot(r1)
plot(r1, resid(.)~fitted(.) | Rail)

- These plots are made by lattice (Trellis) using xyplot
- Considerable freedom in choice of display
ML versus REML

\texttt{lme(travel~1, data=Rail, method="ML")}

- ML estimators generally biased downwards.
- Here, only the estimate for Rail effect is biased
The `gls` function

- Like `lm`, but allows you to specify error model
- Variance functions (`weights` argument)
- Correlation structure (`correlation` argument)
- Similar syntax applies to `lme()`
Variance function classes

- Describe functional dependence (e.g. power function)
- Can depend on parameters and covariates
- Possibly have different parameters for different groups
- **Basic syntax** `varPower(form=~age|sex)`
  - `age` is the *variance covariate*
  - `sex` is a grouping (separate parameters per group)
- **Special covariate** `fitted(.)`
Precooked variance function classes

- **Notice**: `varPower()` creates an object of class "varPower" inheriting from "varFunc"
- `gnls` "knows" how to use such objects and associated methods to fit the model
- **Relevant help page**: `varClasses`
- **Currently** `varExp`, `varPower`, `varConstPower`, `varIdent`, `varFixed`, `varComb`
- **Notice** that they work on *standard deviation scale* (somewhat counterintuitively)
Writing your own

- The internal structure of a variance function object is a parameter vector with a bunch of attributes.
- You’ll need to change the calculation of the "weights" and "logLike" attributes.
- Modify the functions for an existing class, e.g. varPower.
- The functions are fairly complex, but there are actually not many lines to modify.
Correlation structures

► Much the same story as variance functions
► There can be (multiple) covariate dependencies, typically defining spatial coordinates/distances.
► The symbol now means “within group”
► For help, look at corClasses
► corAR1, corARMA, corCAR1, corCompSymm, corExp, corGaus, corLin, corRatio, corSpher, corSymm
► The functions tend to have \texttt{.C()} calls inside, so not too attractive to write your own (but you could...)
Example

- Let us consider the spinal BMD example from yesterday
- To get an unrestricted covariance matrix within subject we need to use
  - an unrestricted correlations structure (`corSymm`)
  - *and* a variance that depends on time
- Notice that we need to use `gls`. Trying to use `lme` just leads to confusion
library(nlme)
spi <- read.csv("spinebmd.csv")
spiLong <- reshape(spi, varying=list(names(spi)[2:8]), v.names="spinebmd", direction="long")

library(lattice)
spiLong$time2 <- c(0,3,6,9,12,18,24)[spiLong$time]
spiLong$grp <- factor(spiLong$grp)
xyplot(spinebmd~time2|id, groups=grp, type="l", data=na.omit(spiLong))
#this takes a while...
ml <- gls(spinebmd~time2+grp:time2, data=na.omit(spiLong), correlation=corSymm(form=~time|id), weight=varIdent(form=~1|time))
summary(ml)
Comparisons between \texttt{gls} and \texttt{lme} models

- Models from the two functions can be compared trivially, using \texttt{anova}

- E.g., let us compare the unrestricted model with a random coefficient regression

```r
m2 <- lme(fixed=spinebmd~time2+grp:time2, random=~time2|id, data=na.omit(spiLong))
anova(m1,m2)
summary(m2)
```
Patterned covariance matrices for random effects

Back with `lme` consider an alternative way of specifying a `Rail` effect:

```r
one <- rep(1, nrow(Rail))
lme(travel ~ 1, random = list(one = pdIdent(~Rail - 1)),
    data = Rail)
```

- Using `one` creates a grouping with only one group
- Notice that we now specify random terms as a list of objects per grouping level
- Random effects have covariance structure according to `pdIdent`, i.e. $\Psi = \tau^2 I$. It is important to use the `-1` in the formula.
The **pdMat** classes

- **pdSymm**: general positive-definite matrix, with no additional structure
- **pdDiag**: diagonal
- **pdIdent**: multiple of an identity
- **pdCompSymm**: compound symmetry structure (constant diagonal and constant off-diagonal elements)
- **pdBlocked**: block-diagonal matrix, with diagonal blocks of any "atomic" ‘pdMat’ class
- Roll your own (rather hard work: each class has 10–17 methods!)
Extended residual variances

- The same error modeling techniques as for gls models are available
- E.g. you can have different variances in each group

lme(travel~1, data=Rail, weights=varIdent(form=~1|Rail))
Another example

```r
data(Orthodont)
OrthoFem <- subset(Orthodont, Sex=="Female")

plot(OrthoFem)
plot(intervals(lmList(distance ~ age, data=OrthoFem)))
```

- Now there’s a covariate `age` involved
- `lmList` calculates individual regression lines
- Notice that the intercepts are not really informative
Comparing models

of1 <- lme(distance ~ age, data=OrthoFem, random=~1|Subject)
of2 <- lme(distance ~ age, data=OrthoFem, random=~age|Subject)
anova(of1,of2)

- Compare random intercept model to random line model
- Likelihood ratio test (on 2 DF)
Plotting results

```
plot(augPred(of2), grid=TRUE, aspect="xy")
```

- The function `augPred` *augments* the model with predicted values.
- Plotting an `augPred` object overlays fitted curves on data.
- `(grid=TRUE plots grey grid, aspect="xy" tries to make average slope 45° to horizontal)
- Notice how lines are biased towards central slope.
A two-level model

data(Pixel)
px1 <- lme(pixel~day+I(day^2), data=Pixel,
         random=list(Dog=~day,Side=~1))
plot(augPred(px1))

- Notice S/R difference: \texttt{I(.)} needed in R
- A 3rd form of random effects: List of formulas
- Fitted curves are parabolic
- Curvature is constant
- Sides of same dog: vertical displacement
- Different dogs may have peaks in different places
Modifying the random structure

```r
px2 <- update(px1, random=~day|Dog)
anova(px1, px2)

px3 <- update(px1, random=~1|Dog/Side)
anova(px1, px3)
```

- Cannot eliminate random effect of `Side`
- Cannot assume all peaks at same position
Comparing fixed terms

```r
px4 <- update(px1, fixed=~.+Side)
summary(px4)
anova(px4)

px1ml <- update(px1, method="ML")
px4ml <- update(px4, method="ML")
anova(px1ml, px4ml)
```

- Note necessary to extend fixed part of model with `Side` term
- No *systematic* effect of side
- Likelihood tests between models do not work with REML
Crossed random effects

- `lme` generally assumes *nested* random effects
- I.e. no particular between person 1 in group 1 and person 1 in group 2
- However, crossed effects can be relevant
- E.g., measurements made with $p$ bloodflow probes simultaneously at $n$ times
- There can be both correlations between measurements at the same time and between measurements with the same probe
Coding crossed effects

```r
data(Assay)
asl <- lme(logDens~sample*dilut, data=Assay,
  random=pdBlocked(list(
    pdIdent(~1),
    pdIdent(~sample-1),
    pdIdent(~dilut-1))))
```

- Notice that we must again use `~sample-1` to “trick” R into generating incidence matrices rather than contrast matrices.
- You lose some of the multilevel structure this way, but not all of it (independent blocks are retained).
Alternate forms

\begin{verbatim}
as2 <- lme(logDens~sample*dilut, data=Assay,
            random=list(Block=pdBlocked(list(
                pdIdent(~1),
                pdIdent(~sample-1))),dilut=~1))
as3 <- lme(logDens~sample*dilut, data=Assay,
            random=list(Block=~1,
                Block=pdIdent(~sample-1),
                dilut=~1))
\end{verbatim}

- None of them gets DF quite right though...
A Growth Curve Example

- Susanne Kjærgaard collected these data on children with CDG (Congenital Deficiency of Glycosylation)
- This disease causes severe growth retardation
- Purpose: Reference curves
- Mean value modeled using spline curve
- Random part as random coefficient regression + autoregression
- Digging out the relevant part of the model object to find the covariance of the random regression terms was a bit tricky
Outline of analysis

(Some plotting code omitted)

lme.obj <- lme(log(Height)~ns(sqrt(Age),
knots=sqrt(c(0.25,.5,1,5)),
   Boundary.knots=sqrt(c(0,10))),
   random=~sqrt(Age)|ID,
   correlation=corExp(form=~sqrt(Age),nugget=F))
Age.new <- seq(0,10,0.01)
C.mat <- lme.obj$sigma^2
   * as.matrix(lme.obj$modelStruct$reStruct$ID)
SD <- sqrt(sapply(Age.new,
   function(a){x<-c(1,sqrt(a)); t(x) %*% C.mat %*% x})
   +lme.obj$sigma^2)