16 - Mixed models with `caracas`

Mikkel Meyer Andersen and Søren Højsgaard

Sun Mar 2 09:30:02 2025

1 Introduction

This vignette is based on `caracas` version 2.1.2.9004. `caracas` is avavailable on CRAN at [https://cran.r-project.org/package=caracas] and on github at [https://github.com/r-cas/caracas].

2 Linear mixed model

A linear mixed model can be written in general form as

$$y = Xb + Zu + e$$

Here, y is a vector of observables, X is a model matrix and b is a corresponding vector of regression coefficients. Also Z is a model matrix and u is a corresponding vector of random effects. Lastly, e is also a vector of random errors. Because there are two random effects (u and e) such models are often called mixed models or variance component models.

It is assumed that u and e are independent and that $u \sim N(0, G)$ and $e \sim N(0, R)$. Typically G and R depend on unknown parameters θ, so we may write $G(\theta)$ and $R(\theta)$ instead.

Consequently,

$$E(y) = \mu = Xb, \quad Var(y) = V(\theta) = ZG(\theta)Z' + R(\theta).$$

As such, linear mixed model, is just a general model for the normal distribution,

$$y \sim N(Xb, V(\theta))$$

where $V = V\theta$ has the special structure given above. In the following

1. We illustrate fitting a mixed model based on a subset of the shoes data available, e.g. in the MASS and doBy packages.
2. We also compare the results with the output from lmer() if the lme4 package is installed and with gls() if the nlme package is installed.

2.1 The likelihood

The log-likelihood is

$$logL(b,\theta) = -\frac{1}{2}(\log|V(\theta)| + (y - Xb)'V(\theta)^{-1}(y - Xb)).$$

For any value of θ, the MLE for b is

$$\hat{b} = \hat{b}(\theta) = (X'V(\theta)^{-1}X)^{-1}X'V(\theta)^{-1}y.$$

Plugging this estimate into the log-likelihood gives the profile log-likelihood which is a function of θ only:

$$plogL(\theta) = -\frac{1}{2}(\log|V(\theta)| + (y - X\hat{b}(\theta))'V(\theta)^{-1}(y - X\hat{b}(\theta))).$$

This function must typically be maximized with numerical methods.

2.2 Shoes data

```
shoes_long <- data.frame(
    stringsAsFactors = FALSE,
    type = c("A", "B", "A", "B", "A", "B", "A", "B"),
    wear = c(13.2, 14, 8.2, 8.8, 10.9, 11.2, 14.3, 14.2),
    boy = as.factor(c("1", "1", "2", "2", "3", "3", "4", "4")))

y <- shoes_long$wear
X <- model.matrix(~type, data=shoes_long) |> head(10)
Z <- model.matrix(~-1+boy, data=shoes_long) |> head(10)
y |> head()
#> [1] 13.2 14.0  8.2  8.8 10.9 11.2
X |> head()
#>   (Intercept) typeB
#> 1           1     0
#> 2           1     1
#> 3           1     0
#> 4           1     1
#> 5           1     0
#> 6           1     1
Z |> head()
#>   boy1 boy2 boy3 boy4
#> 1    1    0    0    0
#> 2    1    0    0    0
#> 3    0    1    0    0
#> 4    0    1    0    0
```

```
#> 5    0    0    1    0
#> 6    0    0    1    0
```

3 Fitting model with `caracas`

We define the following symbols in caracas. All caracas symbols are postfixed with an underscore.

```
y_ <- as_sym(y)
X_ <- as_sym(X)
Z_ <- as_sym(Z)
b_ <- vector_sym(ncol(X_), "b")
def_sym(tau2, sigma2)
```

```
## Covariance matrices for random effects
G_  <- diag_("tau^2", ncol(Z))
R_  <- diag_("sigma^2", nrow(Z))
```

So for this example, $\theta = (\tau, \sigma)$. The variance of y is

```
## Variance etc of y
ZGZt_ <- Z_ %*% G_ %*% t(Z_)
V_  <- ZGZt_ + R_
detV_ <- determinant(V_, log=FALSE)
```

Various methods are available in `caracas` for inverting such a matrix

$$G = \begin{bmatrix} \tau^2 & . & . & . \\ . & \tau^2 & . & . \\ . & . & \tau^2 & . \\ . & . & . & \tau^2 \end{bmatrix}$$

$$R = \begin{bmatrix} \sigma^2 & . & . & . & . & . & . & . \\ . & \sigma^2 & . & . & . & . & . & . \\ . & . & \sigma^2 & . & . & . & . & . \\ . & . & . & \sigma^2 & . & . & . & . \\ . & . & . & . & \sigma^2 & . & . & . \\ . & . & . & . & . & \sigma^2 & . & . \\ . & . & . & . & . & . & \sigma^2 & . \\ . & . & . & . & . & . & . & \sigma^2 \end{bmatrix}$$

$$ZGZ^{\top} = \begin{bmatrix} \tau^2 & \tau^2 & . & . & . & . & . & . \\ \tau^2 & \tau^2 & . & . & . & . & . & . \\ . & . & \tau^2 & \tau^2 & . & . & . & . \\ . & . & \tau^2 & \tau^2 & . & . & . & . \\ . & . & . & . & \tau^2 & \tau^2 & . & . \\ . & . & . & . & \tau^2 & \tau^2 & . & . \\ . & . & . & . & . & . & \tau^2 & \tau^2 \\ . & . & . & . & . & . & \tau^2 & \tau^2 \end{bmatrix}$$

$$V = \begin{bmatrix} \sigma^2 + \tau^2 & \tau^2 & . & . & . & . & . & . \\ \tau^2 & \sigma^2 + \tau^2 & . & . & . & . & . & . \\ . & . & \sigma^2 + \tau^2 & \tau^2 & . & . & . & . \\ . & . & \tau^2 & \sigma^2 + \tau^2 & . & . & . & . \\ . & . & . & . & \sigma^2 + \tau^2 & \tau^2 & . & . \\ . & . & . & . & \tau^2 & \sigma^2 + \tau^2 & . & . \\ . & . & . & . & . & . & \sigma^2 + \tau^2 & \tau^2 \\ . & . & . & . & . & . & \tau^2 & \sigma^2 + \tau^2 \end{bmatrix}$$

$$detV = \left(\sigma^4 + 2\sigma^2\tau^2\right)^4$$

The inverse of a block diagonal matrix is block diagonal. The first blocks of V^{-1} are:

```
Vi_  <- inv_woodbury(R_, Z_, G_) |> simplify() ## or
Vi_  <- inv(V_, method="block") |> simplify()
```

$$V^{-1} = \begin{bmatrix} \frac{\sigma^2+\tau^2}{-\tau^4+\left(\sigma^2+\tau^2\right)^2} & \frac{\tau^2}{\tau^4-\left(\sigma^2+\tau^2\right)^2} & \cdot & \cdot \\ \frac{\tau^2}{\tau^4-\left(\sigma^2+\tau^2\right)^2} & \frac{\sigma^2+\tau^2}{-\tau^4+\left(\sigma^2+\tau^2\right)^2} & \cdot & \cdot \\ \cdot & \cdot & \frac{\sigma^2+\tau^2}{-\tau^4+\left(\sigma^2+\tau^2\right)^2} & \frac{\tau^2}{\tau^4-\left(\sigma^2+\tau^2\right)^2} \\ \cdot & \cdot & \frac{\tau^2}{\tau^4-\left(\sigma^2+\tau^2\right)^2} & \frac{\sigma^2+\tau^2}{-\tau^4+\left(\sigma^2+\tau^2\right)^2} \end{bmatrix}$$

3.1 Programmatic approach

A programmatic approach is to define a function returning the log-likelihood and the profile log-likelihood as a caracas symbol.

```
get_logL0 <- function(y, X, V, b=NULL) {
    if (is.null(b))
        b <- vector_sym(ncol(X_), "b")
    Vi   <- inv(V, method="block")
    detV <- determinant(V, log=FALSE)
    res  <- y - X %*% b
    Q    <- t(res) %*% Vi %*% res
    aux  <- list(b=b, Q=Q)
    out  <- -0.5 * (log(detV) + Q)
    attr(out, "aux") <- aux
    return(out)
}

get_logLp <- function(y, X, V) {
    Vi   <- inv(V, method="block")
    detV <- determinant(V, log=FALSE)
    b    <- solve(t(X) %*% Vi %*% X, t(X) %*% Vi) %*% y
    res  <- y - X %*% b
    Q    <- t(res) %*% Vi %*% res
    aux  <- list(b=b, Q=Q)
    out  <- -0.5 * (log(detV) + Q)
    attr(out, "aux") <- aux
    return(out)
}
```

The following functions also return caracas symbols:

```
get_b_est <- function(y, X, V) {
    Vi <- inv(V, method="block")
    return(solve(t(X) %*% Vi %*% X, t(X) %*% Vi) %*% y)
}

get_V <- function(Z, G, R) {
    return(Z %*% G %*% t(Z) + R)
}
```

```
get_hessian <- function(logL, b) {
    return(-der2(logL, b))
}
```

3.2 Maximizing the profile likelihood

```
V_     <- get_V(Z_, G_, R_)
b_     <- vector_sym(ncol(X_), "b")
b_est_ <- get_b_est(y_, X_, V_) |> simplify()
```

$$b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \hat{b} = \begin{bmatrix} 11.65 \\ \frac{1.60000000000001\sigma^2+3.20000000000002\tau^2}{4.0\sigma^2+8.0\tau^2} \end{bmatrix}$$

Notice that $\hat{b}$ is a function of V and hence of θ. If $\hat{b}$ is substituted into the log-likelihood, we get the profile log-likelihood. On the other hand, b is a parameter of fixed effect and the log-likelihood is a function of both b and θ.

Get logL as a symbol which depends on some variables

```
logLp <- get_logLp(y_, X_, V_)
all_vars(logLp)
#> [1] "tau"   "sigma"

logL0 <- get_logL0(y_, X_, V_)
all_vars(logL0)
#> [1] "b1"    "tau"   "b2"    "sigma"
```

```
out <- optim_sym(c(.1, .1),
                 logLp,
                 method="L-BFGS-B",
                 control=list(fnscale=-1), hessian=TRUE)
par <- out$par
par
#>  sigma    tau
#> 0.2398 2.2677
subs(attr(logLp, "aux")$b, par)
#> c: [11.65  0.4]ᵀ
b_est2_ <- subs(b_est_, par)
b_est2_
#> c: [11.65  0.4000000000000002]ᵀ
as_expr(b_est2_)
#>       [,1]
#> [1,] 11.65
#> [2,]  0.40
```

A technicality: A caracas symbol can be coerced to an R function. Therefor $\hat{b}$ can be evaluated at the MLE as a function of θ.

```
as(b_est_, "function")
#> function (parm, length_parm = 2, names_parm = c("sigma", "tau"))
#> {
#>     sigma = parm[1]
#>     tau = parm[2]
#>     matrix(c(11.65, (1.60000000000001 * sigma^2 + 3.20000000000002 *
#>         tau^2)/(4 * sigma^2 + 8 * tau^2)), nrow = 2)
#> }
#> <environment: 0x583dacb97a48>
as(b_est_, "function")(par)
#>       [,1]
#> [1,] 11.65
#> [2,]  0.40
```

3.3 Asymptotic variance of the MLE

To obtain the asymptotic variance of $\hat{b}$, we need to compute the Hessian of the log-likelihood at the MLE. This is done by differentiating the log-likelihood with respect to b and then evaluating at the MLE. The Hessian is then inverted to obtain the asymptotic variance of $\hat{b}$.

```
logL_ <- get_logL0(y_, X_, V_) |> simplify()
J_    <- get_hessian(logL_, b_)

J2_   <- subs(J_, par)
Vb_   <- solve(J2_)

as_expr(b_est2_)
#>       [,1]
#> [1,] 11.65
#> [2,]  0.40
as_expr(Vb_)
#>          [,1]     [,2]
#> [1,]  1.30000 -0.01438
#> [2,] -0.01438  0.02875
```

3.4 Maximizing the full likelihood

An alternative is to maximize the full likelihood, we need to maximize with respect to both the variance parameters and the fixed effects. For this to work we often need to impose restrictions on the parameter space (not necessary for this specific example, though)

```
eps <- 1e-6
out <- optim_sym(c(0, 0, .1, .1), logL_, method="L-BFGS-B",
```

```
                  lower=c(-Inf, -Inf, eps, eps), upper=c(Inf, Inf, Inf, Inf),
                  control=list(fnscale=-1), hessian=TRUE)
out$par
#>      b1      b2   sigma     tau
#> 11.6500  0.4000  0.2398  2.2677
solve(-out$hessian)[1:2, 1:2] |> round(4)
#>         b1      b2
#> b1  1.2999 -0.0144
#> b2 -0.0144  0.0288
```

4 Comparison with `gls()` and `lmer()`

4.1 Comparison with gls() - if available

```
if (require(nlme)) {
    model <- gls(wear ~ type, correlation = corCompSymm(form = ~ 1 | boy),
                 method="ML",
                 data = shoes_long)

    sigma2 <- model$sigma^2
    C <- corMatrix(model$modelStruct$corStruct)
    V <- sigma2 * as.matrix(bdiag(C))
    as(V, "sparseMatrix")
}
#> Loading required package: nlme
#> 8 x 8 sparse Matrix of class "dsCMatrix"
#>
#> [1,] 5.200 5.143 .     .     .     .     .     .
#> [2,] 5.143 5.200 .     .     .     .     .     .
#> [3,] .     .     5.200 5.143 .     .     .     .
#> [4,] .     .     5.143 5.200 .     .     .     .
#> [5,] .     .     .     .     5.200 5.143 .     .
#> [6,] .     .     .     .     5.143 5.200 .     .
#> [7,] .     .     .     .     .     .     5.200 5.143
#> [8,] .     .     .     .     .     .     5.143 5.200
```

4.2 Comparison with lmer() - if available

```
if (require(lme4)) {
    lmm_fit  <- lmer(wear ~ type + (1|boy), data=shoes_long, REML=FALSE)
    print(VarCorr(lmm_fit))
    print(fixef(lmm_fit))
    print(vcov(lmm_fit))

    var.d <- crossprod(getME(lmm_fit,"Lambdat"))
    Zt <- getME(lmm_fit, "Zt")
```

```
    vr <- sigma(lmm_fit)^2
    var.b <- vr*(t(Zt) %*% var.d %*% Zt)
    sI <- vr * Diagonal(ncol(Zt))
    var.y <- var.b + sI
    var.y
}
#> Loading required package: lme4
#>
#> Attaching package: 'lme4'
#> The following object is masked from 'package:nlme':
#>
#>     lmList
#>  Groups   Name        Std.Dev.
#>  boy      (Intercept) 2.27
#>  Residual             0.24
#> (Intercept)       typeB
#>       11.65        0.40
#> 2 x 2 Matrix of class "dpoMatrix"
#>             (Intercept)    typeB
#> (Intercept)     1.30000 -0.01437
#> typeB          -0.01437  0.02875
#> 8 x 8 sparse Matrix of class "dgCMatrix"
#>       1     2     3     4     5     6     7     8
#> 1 5.200 5.143 .     .     .     .     .     .
#> 2 5.143 5.200 .     .     .     .     .     .
#> 3 .     .     5.200 5.143 .     .     .     .
#> 4 .     .     5.143 5.200 .     .     .     .
#> 5 .     .     .     .     5.200 5.143 .     .
#> 6 .     .     .     .     5.143 5.200 .     .
#> 7 .     .     .     .     .     .     5.200 5.143
#> 8 .     .     .     .     .     .     5.143 5.200
```