What happens here?

Søren Højsgaard

2025-03-02

Table of contents

1 Beyond linear normal models 1

2 Orthodont data 3

```
library(Matrix)
library(caracas)
library(broom)
library(doBy)
library(nlme)
library(lme4)
library(broom.mixed)
```

1 Beyond linear normal models

Recall Orthodont data: Four measurements per child at equidistant time points. There is a population intercept for each gender and a random intercept for each child. Thee is a linear treand over time, and on can consider the idea of modelling a random slope.

This lead to our linear mixed model

$$y = Xb + Zu + e$$

The ultimate goal is to fit a model where there is a random intercept but where the errors are not $e \sim N(0, \sigma^2 I)$ but has an auto regressive structure.

For each child we assume an AR(1) correlation matrix. The correlation matrix for all is a block diagonal matrix with the matrix above on the diagonal. An easy way of creating such a matrix is using Kronecker product. Here for two children

```
bs  <- 4 # Number of measurements per child (block):
nb  <- 2 # Number of blocks (children)
V.b <- toeplitz_(paste0("r^",0:(bs-1)))
I2  <- diag_(1, nb)
V.all <- kronecker(I2, V.b)
## V.b   <- toeplitz(c(1, w, rep(0, bs-2))) ## Perhaps an alternative
```

At some point of time we need to make a numerical evaluation based on as specific value of r. The easiest is to turn R into and R function:

```
V_fn <- as_func(V.all)
V_fn(.1) |> as("sparseMatrix")
```

```
8 x 8 sparse Matrix of class "dsCMatrix"

[1,] 1.000 0.10 0.01 0.001 .     .    .    .
[2,] 0.100 1.00 0.10 0.010 .     .    .    .
[3,] 0.010 0.10 1.00 0.100 .     .    .    .
[4,] 0.001 0.01 0.10 1.000 .     .    .    .
[5,] .     .    .    .     1.000 0.10 0.01 0.001
[6,] .     .    .    .     0.100 1.00 0.10 0.010
[7,] .     .    .    .     0.010 0.10 1.00 0.100
[8,] .     .    .    .     0.001 0.01 0.10 1.000
```

That is, three lines of code using caracas, and we are good to go

$$V.b = \begin{bmatrix} 1 & r & r^2 & r^3 \\ r & 1 & r & r^2 \\ r^2 & r & 1 & r \\ r^3 & r^2 & r & 1 \end{bmatrix}, V.all = \begin{bmatrix} 1 & r & r^2 & r^3 & 0 & 0 & 0 & 0 \\ r & 1 & r & r^2 & 0 & 0 & 0 & 0 \\ r^2 & r & 1 & r & 0 & 0 & 0 & 0 \\ r^3 & r^2 & r & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & r & r^2 & r^3 \\ 0 & 0 & 0 & 0 & r & 1 & r & r^2 \\ 0 & 0 & 0 & 0 & r^2 & r & 1 & r \\ 0 & 0 & 0 & 0 & r^3 & r^2 & r & 1 \end{bmatrix}$$

Of course, we can do the same in "pure R" - but for a specific value of **r**:

```
V_fn2 <- function(r, nb, bs){
    V.b   <- toeplitz(c(r^(0:(bs-1))))
    I     <- diag(1, nb)
    V.all <- kronecker(I, V.b)
    return(V.all)
}

V_fn2(.1, 2, 4) |> as("sparseMatrix")
```

```
8 x 8 sparse Matrix of class "dsCMatrix"

[1,] 1.000 0.10 0.01 0.001 .     .    .    .
[2,] 0.100 1.00 0.10 0.010 .     .    .    .
[3,] 0.010 0.10 1.00 0.100 .     .    .    .
[4,] 0.001 0.01 0.10 1.000 .     .    .    .
[5,] .     .    .    .     1.000 0.10 0.01 0.001
[6,] .     .    .    .     0.100 1.00 0.10 0.010
[7,] .     .    .    .     0.010 0.10 1.00 0.100
[8,] .     .    .    .     0.001 0.01 0.10 1.000
```

```
worker <- function(y, X, V){
    Lt <- chol(V)
    Vi <- chol2inv(Lt)
    ld_V <- 2*sum(log(diag(Lt)))
    n   <- nrow(X)
    b   <- solve(t(X) %*% Vi  %*% X, t(X) %*% Vi %*% y)
    res <- (y - X %*% b)
    Q   <- t(res) %*% Vi %*% res
```

```
    v   <- Q / n
    return(list(b=b, n=n, v=v, ld_V=ld_V, Q=Q))
}

logL.1 <- function(parm, X, y, nb, bs) {
    ## v is estimated from residual sums of squares
    w  <- parm[1] ## rho
    V  <- V_fn2(w, nb, bs)
    aux <- worker(y, X, V)
    obj <- as.numeric(-(aux$n/2) * log(aux$v) - aux$ld_V / 2 - (1/(2*aux$v)) * aux$Q)
    attributes(obj) <- list(aux=aux)
    return(obj)
}

logL.2 <- function(parm, X, y, nb, bs) {
    ## v is multiplied onto V
    w <- parm[1] ## rho
    v <- parm[2] ## sigma^2
    V  <- v * V_fn2(w, nb, bs)
    aux <- worker(y, X, V)
    obj <- as.numeric(- aux$ld_V / 2 - (1/2) * aux$Q)
    attributes(obj) <- list(aux=aux)
    return(obj)
}

logL.3 <- function(parm, X, y, nb, bs) {
    ## v is estimated as separate parameter
    w <- parm[1] ## rho
    v <- parm[2] ## sigma^2
    V  <- V_fn2(w, nb, bs)
    aux <- worker(y, X, V)
    obj <- as.numeric(-(aux$n/2) * log(v) - aux$ld_V / 2 - (1/(2 * v)) * aux$Q)
    attributes(obj) <- list(aux=aux)
    return(obj)
}
```

2 Orthodont data

```
dat <- Orthodont[1:20,]
y <- dat$distance
X <- model.matrix(~1, data=dat)
X
```

```
   (Intercept)
1            1
2            1
3            1
4            1
5            1
6            1
7            1
8            1
9            1
10           1
11           1
12           1
13           1
14           1
15           1
```

```
16           1
17           1
18           1
19           1
20           1
attr(,"assign")
[1] 0
```

```
nb <- unique(dat$Subject) |> length()
bs <- 4

mm <- lmer(distance ~ age + (1|Subject), data=dat)
mm |> tidy()
```

```
# A tibble: 4 x 6
  effect   group    term            estimate std.error statistic
  <chr>    <chr>    <chr>              <dbl>     <dbl>     <dbl>
1 fixed    <NA>     (Intercept)       17.3       1.71      10.1
2 fixed    <NA>     age                0.700     0.131      5.36
3 ran_pars Subject  sd__(Intercept)    1.98     NA         NA
4 ran_pars Residual sd__Observation    1.31     NA         NA
```

```
args <- list(X=X, y=y, nb=nb, bs=bs)
logL.1. <- doBy::set_default(logL.1, args)
logL.2. <- doBy::set_default(logL.2, args)
logL.3. <- doBy::set_default(logL.3, args)

opt.1 <-
    optim(c(0.1),
          logL.1.,
          control = list(fnscale=-1))

opt.2 <-
    optim(c(0, 1),
          logL.2.,
          control = list(fnscale=-1))

opt.3 <-
    optim(c(0, 1),
          logL.3.,
          control = list(fnscale=-1))


source("free_parms.r")

par <- set_free(0.1, list(cor=1))
opt.1 <-
    optim(par,
          logL.1, X=X, y=y, nb=nb, bs=bs,
          control = list(fnscale=-1))

par <- set_free(c(0, 0), list(cor=1, var=2))
opt.2 <-
    optim(par,
          logL.2, X=X, y=y, nb=nb, bs=bs,
          control = list(fnscale=-1))

par <- set_free(c(0, 0), list(cor=1, var=2))
opt.3 <-
    optim(par,
          logL.3, X=X, y=y, nb=nb, bs=bs,
          control = list(fnscale=-1))
```

```
logL.3(opt.3$par, X=X, y=y, nb=nb, bs=bs)
```

```
[1] -26.43
attr(,"aux")
attr(,"aux")$b
             [,1]
(Intercept) 25.19

attr(,"aux")$n
[1] 20

attr(,"aux")$v
     [,1]
[1,] 7.65

attr(,"aux")$ld_V
[1] -7.824

attr(,"aux")$Q
     [,1]
[1,]  153
```

```
args <- list(X=X, y=y, nb=nb, bs=bs)
logL.1. <- doBy::set_default(logL.1, args)
logL.2. <- doBy::set_default(logL.2, args)
logL.3. <- doBy::set_default(logL.3, args)

par.1 <- set_free(0.1, list(cor=1))
par.2 <- set_free(c(0, 0), list(cor=1, var=2))

opt.1 <-
    optim(par.1, logL.1., ,
          control = list(fnscale=-1))

opt.2 <-
    optim(par.2, logL.2.,
          control = list(fnscale=-1))

opt.3 <-
    optim(par.2, logL.3.,
          control = list(fnscale=-1))

opt.1$par
```

```
  corr
0.6377
```

```
opt.2$par
```

```
  corr    var
0.6375 7.6524
```

```
opt.3$par
```

```
  corr    var
0.6375 7.6524
```

```
logL.1.(opt.1$par)
```

```
[1] -26.43
attr(,"aux")
attr(,"aux")$b
             [,1]
(Intercept) 25.19

attr(,"aux")$n
[1] 20

attr(,"aux")$v
      [,1]
[1,] 7.652

attr(,"aux")$ld_V
[1] -7.83

attr(,"aux")$Q
     [,1]
[1,]  153
```

```
logL.2.(opt.2$par)
```

```
[1] -26.43
attr(,"aux")
attr(,"aux")$b
             [,1]
(Intercept) 25.19

attr(,"aux")$n
[1] 20

attr(,"aux")$v
       [,1]
[1,] 0.9997

attr(,"aux")$ld_V
[1] 32.88

attr(,"aux")$Q
      [,1]
[1,] 19.99
```

```
logL.3.(opt.3$par)
```

```
[1] -26.43
attr(,"aux")
attr(,"aux")$b
             [,1]
(Intercept) 25.19

attr(,"aux")$n
[1] 20

attr(,"aux")$v
     [,1]
[1,] 7.65
```

```
attr(,"aux")$ld_V
[1] -7.824

attr(,"aux")$Q
     [,1]
[1,]  153
```