Comparison with other R packages

Brice Ozenne

October 21, 2025

This vignette make connexions between the output of a linear mixed model and well-known tests (t.test, ANCOVA, Pearson's correlation, Bartlett's test, ...). The second part of the vignette make connexion with other R packages.

1 nlme package

The model class obtained with the `lmm` function overlaps the model class of the `lme` and `gls` functions from the nlme package.

```
library(nlme)
```

For instance, the compound symmetry is equivalent to `corCompSymm` correlation structure, or to a random intercept model (when the within subject correlation is positive):

```
eRI.lmm <- lmm(bmd ~ visit*grp, structure = "RE",
               data = calciumL, repetition = ~visit|girl)
eCS.gls <- gls(bmd ~ visit*grp, correlation = corCompSymm(form=~visit|girl),
               data = calciumL, na.action = na.omit)
eCS.lme <- lme(bmd ~ visit*grp, random = ~1|girl,
               data = calciumL, na.action = na.omit)
logLik(eRI.lmm)
logLik(eCS.lme)
logLik(eCS.gls)
```

```
[1] -2297.3
'log Lik.' -2297.3 (df=12)
'log Lik.' -2297.3 (df=12)
```

The estimated random effect also match:

```
range(ranef(eRI.lmm)-ranef(eCS.lme))
```

```
[1] -6.6939e-08  3.1497e-08
```

Unstructured residual covariance matrix can also be obtained with `gls`:

```
eUN.gls <- gls(bmd ~ visit*grp,
               correlation = corSymm(form=~as.numeric(visit)|girl),
               weights = varIdent(form=~1|visit),
               data = calciumL, na.action = na.omit)
eUN.lmm <- lmm(bmd ~ visit*grp, structure = "UN",
               data = calciumL, repetition = ~visit|girl)
logLik(eUN.gls)
logLik(eUN.lmm)
```

```
'log Lik.' -2218.5 (df=25)
[1] -2218.5
```

1.1 lme4 package

The model class obtained with the `lmm` function overlaps the model class of the `lmer` function from the lme4 package.

```
library(lme4)
library(lmerTest)
```

For instance, the compound symmetry is equivalent to a random intercept model (when the within subject correlation is positive):

```
eRI.lmer <- lmer(bmd ~ visit*grp + (1|girl), data = calciumL)
logLik(eRI.lmer)
logLik(eRI.lmm)
```

```
'log Lik.' -2297.3 (df=12)
[1] -2297.3
```

The estimated random effects match:

```
range(ranef(eRI.lmm)-ranef(eRI.lmer)$girl)
```

```
[1] -7.3817e-08  3.4754e-08
```

Nested random effects correspond to block unstructured:

```
calciumLB <- transform(calciumL, baseline = visit==1)

eNRI.lmm <- lmm(bmd ~ visit*grp, structure = RE(~(1|girl/baseline)),
                repetition = ~visit|girl,
                data = calciumLB)
eNRI.lmer <- lmer(bmd ~ visit*grp + (1|girl/baseline),
                  data = calciumLB)
logLik(eNRI.lmer)
logLik(eNRI.lmm)
```

```
'log Lik.' -2282.1 (df=13)
[1] -2282.1
```

And the estimated random effects still match:

```
eRanefNRI.lmm <- ranef(eNRI.lmm, format = "wide")
eRanefNRI.lmer <- ranef(eNRI.lmer)
## girl
range(eRanefNRI.lmm$estimate-eRanefNRI.lmer$girl)
## baseline
eRanefNRI2.lmm <- c(eRanefNRI.lmm$estimate.FALSE,eRanefNRI.lmm$estimate.TRUE)
eRanefNRI2.lmer <- ranef(eNRI.lmer)$`baseline:girl`
range(na.omit(eRanefNRI2.lmm)-eRanefNRI2.lmer)
```

```
[1] -1.2487e-06  1.5182e-06
[1] -2.4547e-06  1.9463e-06
```

An unstructure residual covariance matrix can also be obtained using random slopes:

```
eUN.lmer <- lmer(bmd ~ visit*grp + (0 + visit|girl),
                 data = calciumL,
                 control = lmerControl(check.nobs.vs.nRE = "ignore"))
logLik(eUN.lmer)
logLik(eUN.lmm)
```

```
Advarselsbesked:
I checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv,  :
  Model failed to converge with max|grad| = 0.0101162 (tol = 0.002, component 1)
'log Lik.' -2218.5 (df=26)
[1] -2218.5
```

The uncertainty is quantified in a slightly different way, e.g.:

```
anova(eUN.lmm)
```

```
                Wald F-tests

                statistic        df p.value
mean: visit       111.043 (4, 96.0)  <1e-04 ***
      grp           0.764 (1,109.9)  0.3840
      visit:grp     2.791 (4, 96.5)  0.0305   *
```

is very similar but not identical to:

```
anova(eUN.lmer) ## only the last line is comparable
```

```
Type III Analysis of Variance Table with Satterthwaite's method
          Sum Sq Mean Sq NumDF DenDF F value Pr(>F)
visit      65527   16382     4  96.8  258.07 <2e-16 ***
grp          162     162     1 109.3    2.55   0.11
visit:grp    710     177     4  96.8    2.79   0.03 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It is also possible to fit cross-random effects such as:

```
data("Penicillin")
eCRI.lmer <- lmer(diameter ~ 1 + (1|plate) + (1|sample), Penicillin)
logLik(eCRI.lmer)
```

```
'log Lik.' -165.43 (df=4)
```

using `lmm`:

```
Penicillin$index <- paste(Penicillin$sample,Penicillin$plate,sep=".")
Penicillin$id <- 1

eCRI.lmm <- lmm(diameter ~ 1 + (1|plate) + (1|sample), data = Penicillin)
logLik(eCRI.lmm)
```

```
[1] -165.43
```

Despite `lmm` being significantly slower, the loglikelihood and random effect still match:

```
range(ranef(eCRI.lmm)$estimate-rbind(ranef(eCRI.lmer)$plate,ranef(eCRI.lmer)$sample))
```

```
[1] -4.4050e-07  6.0499e-07
```

1.2 mmrm package

The package `mmrm` is an alternative implementation of mixed models specified via covariance structures:

```
library(mmrm)
e.mmrm <- mmrm(
  formula = FEV1 ~ RACE + SEX + ARMCD * AVISIT + us(AVISIT | USUBJID),
  data = fev_data
)
```

```
mmrm() registered as car::Anova extension
```

It leads nearly identical results compared to `lmm`:

```
e.lmm <- lmm(
  FEV1 ~ RACE + SEX + ARMCD * AVISIT,
  repetition = ~ AVISIT | USUBJID, structure = "UN",
  data = fev_data, type.information = "expected"
)
```

```
logLik(e.mmrm) - logLik(e.lmm)
range(coef(e.mmrm) - coef(e.lmm))
range(vcov(e.mmrm) - vcov(e.lmm))
```

```
[1] -2.5413e-06
[1] -0.00018345  0.00016319
[1] -0.00039810  0.00020542
```

The main differences are:

- `mmrm` uses the expected information matrix to quantify uncertainty instead of the observed information matrix.
- `mmrm` implements the Kenward and Roger method for computing the degrees of freedom and not only the Satterthwaite approximation
- `mmrm` implements different covariance patterns
- `mmrm` is faster and probably more memorry efficient
- `mmrm` has currently fewer post-processing methods (e.g. adjustment multiple comparisons when testing several model parameters).

1.3 emmeans package

To illustrate a key difference between the emmeans package and the `effects.lmm` function we consider an informative and unbalanced group variable:

```
gastricbypassLB$group2 <- gastricbypassLB$weight1>150
```

Since `lmm`:

```
eCS.lmm_2 <- lmm(glucagonAUC ~ visit*group2, repetition =~visit|id, structure = "CS", data =
    gastricbypassLB)
logLik(eCS.lmm_2)
```

```
[1] -315.2
```

we will use the equivalent with the random effect specification:

```
eRI.lmer_2 <- lmer(glucagonAUC ~ visit*group2 + (1|id), data = gastricbypassLB)
logLik(eRI.lmer_2)
```

```
'log Lik.' -315.2 (df=10)
```

While the two models are equivalent, the average outcome output by `effects`:

```
effects(eCS.lmm_2, variable = NULL)
```

```
                Average counterfactual outcome

      estimate    se   df  lower  upper
(t=1)   32.317 4.426 64.3 23.476 41.158
(t=2)   29.653 4.535 65.2 20.598 38.709
(t=3)   77.308 4.535 65.1  68.25 86.366
(t=4)    51.95 4.426 64.3 43.109 60.791
```

substantially differ from the one of emmeans:

```
library(emmeans)
emmeans(eRI.lmer_2, specs=~visit)
```

```
NOTE: Results may be misleading due to involvement in interactions
 visit emmean   SE   df lower.CL upper.CL
 1       33.6 5.53 64.2     22.6     44.7
 2       32.0 5.57 64.4     20.9     43.2
 3       70.0 5.57 64.4     58.9     81.1
 4       47.2 5.53 64.2     36.1     58.2

Results are averaged over the levels of: group2
Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

This is because when averaging over the level of a covariate, emmeans considers *balanced groups*. In the example, the groups are not balanced:

```
table(gastricbypassLB$group2)/NROW(gastricbypassLB)
```

```
FALSE  TRUE
  0.8   0.2
```

Based on the group and timepoint specific means:

```
eCS.elmm_2 <- model.tables(effects(eCS.lmm_2, variable = "group2"))
eCS.elmm_2
```

```
  group2 visit estimate     se     df  lower  upper    p.value
1  FALSE     1   31.430 4.9484 64.349 21.545 41.314 2.4688e-08
2  FALSE     2   28.067 5.0996 65.383 17.884 38.251 6.6737e-07
3  FALSE     3   82.173 5.1008 65.211 71.986 92.359 0.0000e+00
4  FALSE     4   55.126 4.9484 64.349 45.241 65.010 0.0000e+00
5   TRUE     1   35.864 9.8967 64.349 16.095 55.633 5.7374e-04
6   TRUE     2   35.997 9.8967 64.349 16.228 55.766 5.4953e-04
7   TRUE     3   57.848 9.8967 64.349 38.079 77.617 1.8339e-07
8   TRUE     4   39.246 9.8967 64.349 19.477 59.015 1.8651e-04
```

We illustrate the difference:

- emmeans:

```
0.5*eCS.elmm_2[eCS.elmm_2$group2==FALSE,"estimate"]+0.5*eCS.elmm_2[eCS.elmm_2$group2==TRUE,"
    estimate"]
```

```
[1] 33.647 32.032 70.010 47.186
```

- effects:

```
0.8*eCS.elmm_2[eCS.elmm_2$group2==FALSE,"estimate"]+0.2*eCS.elmm_2[eCS.elmm_2$group2==TRUE,"
    estimate"]
```

```
[1] 32.317 29.653 77.308 51.950
```

The "emmeans" approach gives equal "weight" to the expected value of both group:

```
mu.group1 <-  as.double(coef(e.group)["(Intercept)"])
mu.group2 <-  as.double(coef(e.group)["(Intercept)"] + coef(e.group)["group2TRUE"])
p.group1 <- 14/20          ; p.group2 <- 6/20
c(emmeans = (mu.group1+mu.group2)/2, predict = mu.group1 * p.group1 + mu.group2 * p.group2)
```

```
 emmeans  predict
4.450435 4.514352
```

1.4 effectsize package (R^2 or η^2)

Partial η^2 can be computed based on `lmer` using the effectsize package:

```
library(effectsize)
eta_squared(eCS.lmer)
cat("\n")
```

```
# Effect Size for ANOVA (Type III)

Parameter   | Eta2 (partial) |       95% CI
-------------------------------------------
visit       |           0.64 | [0.50, 1.00]
group       |           0.01 | [0.00, 1.00]
visit:group |           0.19 | [0.03, 1.00]

- One-sided CIs: upper bound fixed at
```

and are approximately equal to what one can compute "manually":

```
eCS.Wald <- anova(eCS.lmm)$multivariate
eCS.Wald$df.num*eCS.Wald$statistic/(eCS.Wald$df.num*eCS.Wald$statistic+eCS.Wald$df.denom)
```

```
[1] 0.335374 0.033811 0.186290
```

The will not be true for heteroschedastic models:

```
eUN.Wald <- anova(eUN.lmm)$multivariate
eUN.Wald$df.num*eUN.Wald$statistic/(eUN.Wald$df.num*eUN.Wald$statistic+eUN.Wald$df.denom)
```

```
[1] 0.50787 0.17905 0.32380
```

compared to:

```
eta_squared(eUN.lmer)
cat("\n")
```

```
# Effect Size for ANOVA (Type III)

Parameter   | Eta2 (partial) |       95% CI
-------------------------------------------
visit       |           0.76 | [0.54, 1.00]
group       |           0.01 | [0.00, 1.00]
visit:group |           0.32 | [0.00, 1.00]

- One-sided CIs: upper bound fixed at
```

But in that case both may be misleading as the proportion of explained variance is not clearly defined.

1.5 MuMIn package (R^2)

```
library(MuMIn)
r.squaredGLMM(eCS.lmer)
cat("\n")
```

```
         R2m     R2c
[1,] 0.51728 0.62222
```

To reproduce these R2, we extract from the random intercept model:

- the residual variance

```
sigmaW <- sigma(eCS.lmm)[1,1]-sigma(eCS.lmm)[1,2]
```

- the variance of the random effect

```
sigmaB <- sigma(eCS.lmm)[1,2]
```

- the variance of the fitted values:

```
sigma2_XB <- var(fitted(eCS.lmm))
```

and evalutae the ratios:

```
c(R2m = sigma2_XB/(sigmaW + sigmaB + sigma2_XB),
  R2c = (sigma2_XB + sigmaB)/(sigmaW + sigmaB + sigma2_XB))
```

```
    R2m     R2c
0.52549 0.62865
```

1.6 stats package (partial residuals)

The function `residuals.lm` can be used to extract partial residuals from `lm` objects. For instance:

```
gastricbypassW$group <- as.factor(as.numeric(gastricbypassW$id)%%2)
eIID.lm <- lm(weight4 ~ group + weight1, data = gastricbypassW)
pRes.lm <- residuals(eIID.lm, type = "partial")
head(pRes.lm)
```

```
      group  weight1
1   7.19282   3.6648
2  -0.20504  31.7052
3   0.60631 -17.3352
4   6.44389  22.7052
5  -1.59403 -16.7352
6 -18.23382   8.4052
```

Those generally differ (by a constant) from the one provided by `residuals.lmm`:

```
eIID.lmm <- lmm(weight4 ~ group + weight1, data = gastricbypassW)
(residuals(eIID.lmm, type = "partial", variable = "group") - pRes.lm[,"group"])
(residuals(eIID.lmm, type = "partial", variable = "weight1") - pRes.lm[,"weight1"])
```

```
     1      2      3      4      5      6      7      8      9     10     11     12     13     14
2.0702 2.0702 2.0702 2.0702 2.0702 2.0702 2.0702 2.0702 2.0702 2.0702 2.0702 2.0702 2.0702 2.0702
    15     16     17     18     19     20
2.0702 2.0702 2.0702 2.0702 2.0702 2.0702
     1      2      3      4      5      6      7      8      9     10     11     12     13     14
106.22 106.22 106.22 106.22 106.22 106.22 106.22 106.22 106.22 106.22 106.22 106.22 106.22 106.22
    15     16     17     18     19     20
106.22 106.22 106.22 106.22 106.22 106.22
```

Indeed, `residuals.lm` centers the design matrix of the variable relative to which the partial residuals are computed:

```
coef(eIID.lm)["group1"] * mean(gastricbypassW$group=="1")
coef(eIID.lm)["weight1"] * mean(gastricbypassW$weight1)
```

```
group1
2.0702
weight1
 106.22
```

For continuous variable with a linear effect, these residuals can be obtained by setting the `type` argument to `"partial-center"`:

```
(residuals(eIID.lmm, type = "partial-center", variable = "weight1") - pRes.lm[,"weight1"])
```

```
          1           2           3           4           5           6           7           8
 1.7675e-13  6.7502e-14 -6.3949e-14  5.6843e-14 -3.9080e-14  8.1712e-14 -3.7303e-14  5.9508e-14
          9          10          11          12          13          14          15          16
-4.2633e-14  4.4409e-14 -2.9310e-14  5.5123e-14 -4.6185e-14  4.4409e-14 -4.2633e-14  4.6185e-14
         17          18          19          20
-3.9968e-14  5.3291e-14 -1.4211e-14  3.5527e-14
```

⚠ When evaluating the partial residuals relative to categorical variables, interactions, or non-linear terms, the output obtained with `partial-center` will not match the one of `residuals.lm`. Indeed `partial-center` will, when numeric, center the original variable whereas `residuals.lm` will center the column relative to the coefficient in the design matrix.

References