Comparison with traditional tests and other R packages

Brice Ozenne

October 16, 2025

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

1 Illustrative datasets

We will consider two illustrative datasets:

- data from the abeta study lifestyle and psychosicial data between patients with newly diagnosed bipolar disorder (`BD`) and matched healthy controls (`HC`) at baseline: functioning assessment test (`fast0`), quality of life (`qol0`), perceived stress score (`pss0`), ... and at 1 year follow-up (`pss1`, `fast1`, `qol1`, for `BD` only).

```
data(abetaW, package = "LMMstar")
abetaW$missingreason <- NULL
head(abetaW)
```

```
  id sex age group episode fast0 qol0 pss0 fast1 pss1 qol1 educationyears alcohol
1  1   M  30    BD       0     1   88    9     0   NA   NA             13       0
2  2   F  55    BD       1    32   87   21    NA   NA   NA             15       0
3  3   M  51    BD       0    29   86   23    31   27   79             21       1
4  4   M  38    BD       0     1   96    7     6    6  101             21       1
5  5   M  21    BD       0     3   97    1     1    5  105             12       1
6  6   M  42    BD       1    22   70   17    40   18   68             13       0
```

This dataset shows great difference in heterogeneity between the two groups, e.g.:

```
library(LMMstar)
summarize(pss0 ~ group, data = abetaW, na.rm = TRUE)
```

```
  group observed missing    mean     sd min   q1 median    q3 max
1    BD       86       1 13.2674 6.8435   1 7.25     13 17.75  29
2    HC       44       0  7.2727 5.0272   0 3.75      6 10.50  19
```

We will also use a balanced version of this dataset (equal group size):

```
abetaW.B <- do.call(rbind, by(abetaW, abetaW$group, function(iDF){
  iDF[which(!is.na(iDF$pss0))[1:44],]
}))
```

- data from the calcium dataset, a two-arm randomized clinical trial comparing bone mineral density between calcium supplement (`C`) and placebo (`P`). Visits were planned every 6 months, `bmd1` refers to the baseline measurement and `bmd2`, ..., `bmd5` refers to post-intervention measurements. `time.obs1`,..., `time.obs5` refer to the time elpased from baseline measurement in years.

```
data(calciumL, package = "LMMstar")
calciumL <- merge(by = "girl", calciumL,
                  transform(calciumL, baseline = bmd)[calciumL$visit==1,c("girl","baseline")])
calciumL <- calciumL[order(calciumL$girl,calciumL$visit),]
head(calciumL)
```

```
   girl grp dropout dropvisit visit time.obs bmd baseline
1   101   C       0        NA     1  0.00000 815      815
3   101   C       0        NA     2  0.51472 875      815
5   101   C       0        NA     3  0.98015 911      815
2   101   C       0        NA     4  1.49760 952      815
4   101   C       0        NA     5  1.99589 970      815
10  102   P       0        NA     1  0.00000 813      813
```

The corresponding wide format is

```
data(calciumW, package = "LMMstar")
calciumW$dropout <- NULL
calciumW$dropvisit <- NULL
head(calciumW)
```

```
  girl grp bmd1 bmd2 bmd3 bmd4 bmd5 time.obs1 time.obs2 time.obs3 time.obs4 time.obs5
1  101   C  815  875  911  952  970         0   0.51472   0.98015    1.4976    1.9959
2  102   P  813  833  855  881  901         0   0.51472   0.95551    1.4730    1.9521
3  103   P  812  812  843  855  895         0   0.51198   0.95825    1.4757    1.9548
4  104   C  804  847  885  920  948         0   0.51198   0.97194    1.5086    2.1136
5  105   C  904  927  952  955 1002         0   0.57495   0.97741    1.4757    1.9548
6  106   P  831  855  890  908  933         0   0.53388   1.01300    1.5907    2.1684
```

We will use the placebo group as reference:

```
calciumW$grp <- relevel(calciumW$grp, "P")
calciumL$grp <- relevel(calciumL$grp, "P")
```

and the change from baseline in bone mineral density:

```
calciumW$change2 <- calciumW$bmd2 - calciumW$bmd1
calciumW$change3 <- calciumW$bmd3 - calciumW$bmd1
calciumW$change4 <- calciumW$bmd4 - calciumW$bmd1
calciumW$change5 <- calciumW$bmd5 - calciumW$bmd1
calciumL$change <- calciumL$bmd - calciumL$baseline
```

For illustrative purpose, we will restrict both dataset to subjects with complete data:

```
calciumW.NNA <- calciumW[rowSums(is.na(calciumW))==0,]
calciumL.NNA <- calciumL[calciumL$girl %in% calciumW.NNA$girl,]
```

as the aim is to show equivalence between statistical tests when there is no missing data.

2 Equivalence with other statistical methods

2.1 Welch two sample t-test

A two sample t-test:

```
with(abetaW, t.test(x = pss0[group=="BD"], y = pss0[group=="HC"]))
```

```
        Welch Two Sample t-test

data:  pss0[group == "BD"] and pss0[group == "HC"]
t = 5.67, df = 112, p-value = 1.1e-07
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 3.8988 8.0906
sample estimates:
mean of x mean of y
  13.2674    7.2727
```

is equivalent to a linear regression with a group-specific residual variance:

```
abetaW$group <- relevel(abetaW$group,"HC")
e.ttest <- lmm(pss0 ~ group, structure = IND(~group),
               data = abetaW, trace = FALSE)
model.tables(e.ttest, effects = "all")
```

```
            estimate      se      df  lower  upper    p.value
(Intercept)   7.2727 0.75788  43.009 5.7443 8.8011 2.9650e-12
groupBD       5.9947 1.05781 112.201 3.8988 8.0906 1.1399e-07
sigma         5.0272 0.54210  43.009 4.0447 6.2484         NA
k.BD          1.3613 0.18014  86.351 1.0464 1.7709 2.2090e-02
```

For comparison a linear model would estimate different standard errors, degrees of freedom, and p-values:

```
model.tables(lmm(pss0 ~ group, data = abetaW))
```

```
            estimate      se     df  lower  upper    p.value
(Intercept)   7.2727 0.94857 128.03 5.3958 9.1496 3.8629e-12
groupBD       5.9947 1.16625 128.03 3.6871 8.3023 1.0000e-06
```

as it does not account for heteroschedasticity. This makes the 'heteroschedastic linear regression' `e.ttest` a natural extension of the t-test when it comes to account for covariates.

In the special case of two groups of equal size, the standard errors will be estimated the same:

```
model.tables(lmm(pss0 ~ group, structure = IND(~group),
                 data = abetaW.B, trace = FALSE))
```

```
            estimate      se     df   lower   upper    p.value
(Intercept)  11.8636 0.98648 43.009  9.8742 13.8530 2.4425e-15
groupHC      -4.5909 1.24399 80.661 -7.0662 -2.1156 4.0523e-04
```

```
model.tables(lmm(pss0 ~ group, data = abetaW.B))
```

```
            estimate      se     df   lower   upper    p.value
(Intercept)  11.8636 0.87964 86.017 10.1150 13.6123 0.00000000
groupHC      -4.5909 1.24399 86.017 -7.0639 -2.1179 0.00039184
```

leading to very similar p-values (degrees of freedom differ slightly).

2.2 Paired t-test

With complete data, a paired t-test:

```
t.test(calciumW.NNA$bmd2, calciumW.NNA$bmd1, paired = TRUE)
```

```
        Paired t-test

data:  calciumW.NNA$bmd2 and calciumW.NNA$bmd1
t = 13, df = 90, p-value <2e-16
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 20.229 27.529
sample estimates:
mean difference
         23.879
```

is equivalent to a LMM with an unstructured covariate pattern:

```
e.lmm2tt <- lmm(bmd ~ visit, repetition = ~visit|girl, structure = "UN",
                data = calciumL.NNA)
model.tables(e.lmm2tt)["visit2",,drop=FALSE]
```

```
       estimate     se     df  lower  upper p.value
visit2   23.879 1.8371 89.968 20.229 27.529       0
```

2.3 Comparing change

2.3.1 Using a Welch two sample t-test

With complete data, a two sample t-test comparing the change from baseline:

```
ttc <- with(calciumW.NNA, t.test(x = change2[grp=="C"], y = change2[grp=="P"]))
ttc
```

```
        Welch Two Sample t-test

data:  change2[grp == "C"] and change2[grp == "P"]
t = 2.03, df = 88.8, p-value = 0.046
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  0.14074 14.49659
sample estimates:
mean of x mean of y
   27.659    20.340
```

is equivalent to a LMM with a stratified unstructured covariate pattern:

```
e.lmm2tt2 <- lmm(bmd ~ visit*grp, repetition = ~visit|girl, structure = UN(~grp),
                 data = calciumL.NNA)
model.tables(e.lmm2tt2)[c("visit2","visit2:grpC"),,drop=FALSE]
```

```
            estimate     se     df    lower  upper    p.value
visit2       20.3404 2.5338 46.005 15.24013 25.441 2.6911e-10
visit2:grpC   7.3187 3.6124 88.734  0.14069 14.497 4.5767e-02
```

The estimate and standard error are exactly the same:

```
c(ttc$estimate["mean of x"] - ttc$estimate["mean of y"],
  se = ttc$stderr)
```

```
mean of x        se
   7.3187    3.6124
```

The only (small) difference lies in the estimation of the degrees of freedom.

2.3.2 Using a linear regression

Using a linear model to compare change over time:

```
eLM.change <- lm(change2 ~ grp, data = calciumW.NNA)
summary(eLM.change)$coef
```

```
            Estimate Std. Error t value   Pr(>|t|)
(Intercept)  20.3404     2.5133  8.0931 2.7975e-12
grpC          7.3187     3.6144  2.0249 4.5878e-02
```

is equivalent to the following mixed model:

```
eLMM.change <- lmm(bmd ~ visit*grp,
                   repetition =~ visit|girl, structure = UN,
                   data = calciumL.NNA)
model.tables(eLMM.change)[c("visit2","visit2:grpC"),]
```

```
            estimate     se     df    lower  upper    p.value
visit2       20.3404 2.5133 88.962 15.34654 25.334 2.8044e-12
visit2:grpC   7.3187 3.6144 88.962  0.13688 14.500 4.5880e-02
```

Here, since the linear regression assumes the same variance in both groups, we did not stratified the covariance pattern on group. The same equivalence would hold with a continuous exposure (say dose) instead of a binary exposure (here `grp`).

In presence of a covariate:

```
calciumW2.NNA <- cbind(calciumW.NNA,
                       age = round(runif(NROW(calciumW.NNA), min = 18, max = 60)))
calciumL2.NNA <- merge(calciumL.NNA, calciumW2.NNA[,c("girl","age")], by = "girl")

eLMadj.change <- lm(change2 ~ grp + age, data = calciumW2.NNA)
summary(eLMadj.change)$coef
```

```
             Estimate Std. Error  t value  Pr(>|t|)
(Intercept) 21.665002    6.66811  3.24905 0.0016405
grpC         7.340294    3.63533  2.01916 0.0465143
age         -0.033579    0.15643 -0.21465 0.8305347
```

one should specify interaction with time in the mixed model to retrieve the same results:

```
eLMMadj.change <- lmm(bmd ~ visit*grp + visit*age,
                      repetition =~ visit|girl, structure = UN,
                      data = calciumL2.NNA)
model.tables(eLMMadj.change)[c("visit2","visit2:grpC"),]
```

```
            estimate     se     df  lower  upper   p.value
visit2       21.6650 6.6681 87.963 8.4135 34.917 0.0016408
visit2:grpC   7.3403 3.6353 87.963 0.1158 14.565 0.0465156
```

2.4 Multiple Student's t-test

To adjust several t-tests for multiple testing, one can use the equivalence with `lmm`. This however require to specify the structure of the data (via the argument `repetition`), i.e., at which level replicates are independent so the software can deduce the appropriate number of independent observation across t-tests:

```
e.ttest2 <- lmm(change2 ~ grp, structure = IND(~grp),
                data = calciumW, repetition = ~1|girl, trace = FALSE)
e.ttest3 <- lmm(change3 ~ grp, structure = IND(~grp),
                data = calciumW, repetition = ~1|girl, trace = FALSE)
e.ttest4 <- lmm(change4 ~ grp, structure = IND(~grp),
                data = calciumW, repetition = ~1|girl, trace = FALSE)
e.ttest5 <- lmm(change5 ~ grp, structure = IND(~grp),
                data = calciumW, repetition = ~1|girl, trace = FALSE)
```

The `anova` method is then used to specify the parameter of interest and the results combined using `rbind`:

```
e.mttest <- rbind(anova(e.ttest2, effects = "grpC=0"),
                  anova(e.ttest3, effects = "grpC=0"),
                  anova(e.ttest4, effects = "grpC=0"),
                  anova(e.ttest5, effects = "grpC=0"))
model.tables(e.mttest, method = "single-step2")
```

```
                estimate     se      df   lower  upper  p.value
change2: grpC=0   6.7507 3.3549 103.014 -1.2205 14.722 0.111849
change3: grpC=0  13.8150 4.8336  95.812  2.3302 25.300 0.014600
change4: grpC=0  12.5190 5.8369  86.835 -1.3497 26.388 0.084579
change5: grpC=0  19.0155 6.4666  86.916  3.6506 34.380 0.011510
```

Note: the `single-step2` adjustment is similar to the `single-step` adjustment of the multcomp package, i.e., a max test adjustment. But instead of relying on the density of a multivariate Student's t-distribution, which requires equal degrees of freedom, it samples in a multivariate distribution with Student's t marginal possibly based on different degrees of freedom and a Gaussian copula. Being based on random sampling, results will slightly change everytime the code is run unless the inital state of the random number generator is set to a specific value before running the code:

```
set.seed(1)
model.tables(e.mttest, method = "single-step2")
```

```
                estimate     se      df   lower  upper  p.value
change2: grpC=0   6.7507 3.3549 103.014 -1.2151 14.717 0.113439
change3: grpC=0  13.8150 4.8336  95.812  2.3379 25.292 0.014590
change4: grpC=0  12.5190 5.8369  86.835 -1.3404 26.378 0.085339
change5: grpC=0  19.0155 6.4666  86.916  3.6609 34.370 0.011640
```

The `LMMstar.options` function can be used to output the number of samples used:

```
LMMstar.options()$n.sampleCopula
```

```
[1] 1e+05
```

and change it:

```
LMMstar.options(n.sampleCopula = 1e5)
```

This whole procedure can be streamlined using the long format and the `mlmm` function:

- the argument `by` indicates how to split the data. A separate model is fitted on each split.
- the argument `effects` indicates the test to be extracted for each model.
- the argument `name.short` is a cosmetic argument: should the name of each test be the covariate value or a combination of the covariate variable and the covariate value?

```
e.mttest2 <- mlmm(change ~ grp, structure = IND(~grp), repetition = ~visit|girl,
                  data = calciumL[calciumL$visit!=1,], trace = FALSE,
                  by = "visit", effects = "grpC=0", name.short = FALSE)
set.seed(1)
model.tables(e.mttest, method = "single-step2")
```

```
                estimate     se      df   lower  upper  p.value
change2: grpC=0   6.7507 3.3549 103.014 -1.2151 14.717 0.113439
change3: grpC=0  13.8150 4.8336  95.812  2.3379 25.292 0.014590
change4: grpC=0  12.5190 5.8369  86.835 -1.3404 26.378 0.085339
change5: grpC=0  19.0155 6.4666  86.916  3.6609 34.370 0.011640
```

The function `mlmm` can be used not only to emulate multiple t-tests but also for multiple linear regressions or linear mixed models. In the special case of multiple Welch two-sample test, a dedicated function `mt.test` offers a more user friendly interface:

```
set.seed(1)
mt.test(change2 + change3 + change4 + change5 ~ grp, data = calciumW)
```

```
Argument 'data' contains 59 missing values.
        estimate     se      df   lower  upper  p.value
change2   6.7507 3.3549 103.014 -1.2151 14.717 0.113439
change3  13.8150 4.8336  95.812  2.3379 25.292 0.014590
change4  12.5190 5.8369  86.835 -1.3404 26.378 0.085339
change5  19.0155 6.4666  86.916  3.6609 34.370 0.011640
```

2.5 ANCOVA

Instead of comparing the final value or the change between groups using a Welch two sample t-test, the ANCOVA is often refered to as the superior approach to assess a treatment effect (Vickers and Altman, 2001). It regresses the group variable and the baseline value against the change:

```
model.tables(lmm(change2 ~ bmd1 + grp, data = calciumW.NNA))
```

```
              estimate        se     df      lower    upper  p.value
(Intercept) -25.742684 25.757918 88.018 -76.930991 25.44562 0.320337
bmd1          0.052948  0.029457 88.018  -0.005592  0.11149 0.075693
grpC          6.741021  3.584377 88.018  -0.382155 13.86420 0.063324
```

or the final value:

```
model.tables(lmm(bmd2 ~ bmd1 + grp, data = calciumW.NNA))
```

```
            estimate        se     df     lower   upper  p.value
(Intercept) -25.7427 25.757918 88.018 -76.93099 25.4456 0.320337
bmd1          1.0529  0.029457 88.018   0.99441  1.1115 0.000000
grpC          6.7410  3.584377 88.018  -0.38215 13.8642 0.063324
```

both leading to equivalent result. The corresponding mixed model constrains the both group to take the same baseline value. This can be specified by introducing a new covariate that only differ between groups after baseline:

```
calciumL.NNA$trt <- ifelse(calciumL.NNA$visit==1,"P",as.character(calciumL.NNA$grp))
calciumL.NNA$trt <- factor(calciumL.NNA$trt, levels = c("P","C"))
ftable(grp = calciumL.NNA$grp, trt = calciumL.NNA$trt, visit = calciumL.NNA$visit)
```

```
        visit  1  2  3  4  5
grp trt
P   P         47 47 47 47 47
    C          0  0  0  0  0
C   P         44  0  0  0  0
    C          0 44 44 44 44
```

We then retrieve the same estimate and similar (but not identical) standard errors and p-values with the following mixed model:

```
e.lmmANCOVA <- lmm(bmd ~ visit*trt, repetition = ~visit|girl, structure = UN,
                   data = calciumL.NNA)
model.tables(e.lmmANCOVA)["visit2:trtC",,drop=FALSE]
```

```
Constant values in the design matrix for the mean structure.
Coefficient "trtC" relative to interaction "visit:trt" has been removed.
            estimate     se     df   lower  upper  p.value
visit2:trtC    6.741 3.5642 88.853 -0.3411 13.823 0.061839
```

To avoid the message about the design matrix, one should 'manually' define the interaction terms:

```
calciumL.NNA$visit.trt <- ifelse(calciumL.NNA$trt == "C", calciumL.NNA$visit, "baseline")
calciumL.NNA$visit.trt <- factor(calciumL.NNA$visit.trt, levels = c("baseline",2:5))
ftable(grp = calciumL.NNA$grp, visit.trt = calciumL.NNA$visit.trt, visit = calciumL.NNA$visit)
```

```
              visit  1  2  3  4  5
grp visit.trt
P   baseline        47 47 47 47 47
    2                0  0  0  0  0
    3                0  0  0  0  0
    4                0  0  0  0  0
    5                0  0  0  0  0
C   baseline        44  0  0  0  0
    2                0 44  0  0  0
    3                0  0 44  0  0
    4                0  0  0 44  0
    5                0  0  0  0 44
```

```
e.lmmANCOVA2 <- lmm(bmd ~ visit + visit.trt, repetition = ~visit|girl, structure = UN,
                    data = calciumL.NNA)
model.tables(e.lmmANCOVA2)["visit.trt2",,drop=FALSE]
```

```
           estimate     se     df   lower  upper  p.value
visit.trt2    6.741 3.5642 88.853 -0.3411 13.823 0.061839
```

As before, in presence of a covariate:

```
summary(lm(bmd2 ~ bmd1 + grp + age, data = calciumW2.NNA))$coef
```

```
             Estimate Std. Error  t value   Pr(>|t|)
(Intercept) -24.26195  26.312105 -0.92208 3.5904e-01
bmd1          1.05346   0.029654 35.52538 1.4385e-53
grpC          6.76689   3.603786  1.87772 6.3770e-02
age          -0.04884   0.154702 -0.31571 7.5298e-01
```

one should add the covariate along with time interactions to retrieve the same estimate and similar standard error/p-value/confindence intervals with a linear mixed model:

```
calciumL2.NNA$visit.trt <- ifelse(calciumL2.NNA$grp == "C", calciumL.NNA$visit, "1")

e.lmmANCOVAadj <- lmm(bmd ~ visit + visit.trt + visit*age, repetition = ~visit|girl,
                      structure = UN, data = calciumL2.NNA)
model.tables(e.lmmANCOVAadj)["visit.trt2",,drop=FALSE]
```

```
           estimate     se     df    lower  upper  p.value
visit.trt2   6.7669 3.5833 87.854 -0.35423 13.888 0.062262
```

A natural extension of the ANCOVA would be to relax the assumption of common residual variance between the two treatment groups:

```
model.tables(lmm(change2 ~ bmd1 + grp, data = calciumW.NNA, structure = IND(~grp)))
```

```
              estimate        se     df       lower    upper  p.value
(Intercept) -25.833272 25.805339 83.926 -77.1506784 25.48413 0.319665
bmd1          0.053052  0.029513 84.179  -0.0056359  0.11174 0.075828
grpC          6.739886  3.585265 87.584  -0.3855470 13.86532 0.063448
```

However the 'straightforward' connexion with mixed model seems lost:

```
e.lmmHANCOVA <- lmm(bmd ~ visit + visit.trt, repetition = ~visit|girl, structure = UN(~grp),
                    data = calciumL.NNA)
model.tables(e.lmmHANCOVA)["visit.trt2",,drop=FALSE]
```

```
           estimate     se     df    lower  upper  p.value
visit.trt2   6.7516 3.5654 88.326 -0.33341 13.837 0.061542
```

2.6 Person's correlation

One can retrieve Pearson's correlation:

```
cor.test(calciumW.NNA$bmd1,calciumW.NNA$bmd5)
```

```
        Pearson's product-moment correlation

data:  calciumW.NNA$bmd1 and calciumW.NNA$bmd5
t = 18.3, df = 89, p-value <2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.83615 0.92551
sample estimates:
    cor
0.88901
```

using a linear mixed model moving to the long format and using an unstructured mean and covariance pattern over time:

```
eCor.lmm <- lmm(bmd ~ visit, repetition = ~visit|girl,
                structure = UN, data = calciumL.NNA)
model.tables(eCor.lmm,  effects = "correlation")["rho(1,5)",]
```

```
         estimate     se     df   lower   upper p.value
rho(1,5)  0.88901 0.0221 96.839 0.83607 0.92555       0
```

P-value and confidence interval will differ (only slightly in large samples) because `cor.test` uses an exact[1] formula for the variance after `atanh` transformation while the linear mixed model rely on the observed information matrix. In this example the observed information (default option) is more in line with `cor.test` than the expected information:

```
model.tables(eCor.lmm,  type.information = "expected", effects = "correlation")["rho(1,5)",]
```

```
         estimate       se       df   lower   upper p.value
rho(1,5)  0.88901 0.021914 17285033 0.83738 0.92492       0
```

Of note the confidence intervals and p-value of `cor.test` are not computed in a consistent way:

```
set.seed(7303)
X <- rnorm(10)
Y <- rnorm(10)
cor.test(X,Y)
```

[1] assuming jointly normally distributed outcomes

```
        Pearson's product-moment correlation

data:  X and Y
t = 2.29, df = 8, p-value = 0.051
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.00016154 0.90179629
sample estimates:
    cor
0.62972
```

Here the confidence intervals do not overlap 0, i.e., suggest to reject the null hypothesis while the p-value is greater than 0.05, i.e., does not suggest to reject the null hypothesis. The corresponding mixed model estimate:

```
dfXY <- rbind(data.frame(value = X, variable = "x", id = 1:10),
              data.frame(value = Y, variable = "y", id = 1:10))
e.lmmXY <- lmm(value ~ variable, repetition = ~variable|id,
               structure = UN, data = dfXY)
model.tables(e.lmmXY, effects = "correlation")
```

```
         estimate      se     df     lower   upper  p.value
rho(x,y)  0.62972 0.20115 7.0024 -0.047159 0.91027 0.061602
```

is the same but the confidence intervals and p-value differ more substantially (due to small sample approximations). They however are consistent with respect to whether to reject the null hypothesis.

2.7 Comparing Person's correlation

To compare the Pearson's correlation between two groups,

```
library(cocor)
cocor.indep.groups()
```

```
eCor2.lmm <- lmm(bmd ~ visit*grp, repetition = ~visit|girl,
                 structure = UN(~grp), data = calciumL.NNA)
model.tables(eCor2.lmm,  effects = "correlation")[c("rho(1,5):C","rho(1,5):P"),]
```

```
           estimate       se     df   lower   upper    p.value
rho(1,5):C  0.85965 0.039801 42.111 0.75492 0.92163 1.2128e-10
rho(1,5):P  0.91701 0.023456 53.835 0.85496 0.95319 7.3275e-15
```

```
summary(anova(eCor2.lmm, effects = "rho(1,5):C - rho(1,5):P = 0", transform.rho = "none"))
```

```
                Multivariate Wald test

       statistic      df p.value
   all     1.542 (1,3.7)   0.288
   ------------------------------
    :  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1.
  df: Satterthwaite approximation w.r.t. model-based se.

                Univariate Wald test

                              estimate    se  df lower upper p.value
   rho(1,5):C - rho(1,5):P = 0   -0.057 0.046 3.7 -0.19 0.076   0.288
   ---------------------------------------------------------------------
    :  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1.
  df: Satterthwaite approximation w.r.t. model-based se.
  se: Modeled based on the observed information.
```

2.8 Correlation between changes

In some studies, one is interested in studying the relation between two evolutions. Say weight and glucagon before and after the operation:

```
gastricbypassW$changeG41 <- gastricbypassW$glucagonAUC4-gastricbypassW$glucagonAUC1
gastricbypassW$changeW41 <- gastricbypassW$weight4-gastricbypassW$weight1
```

One can evaluate their correlation:

```
cor.test(gastricbypassW$changeW41, gastricbypassW$changeG41)
```

```
        Pearson's product-moment correlation

data:  gastricbypassW$changeW41 and gastricbypassW$changeG41
t = 1.89, df = 18, p-value = 0.075
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.043829  0.719624
sample estimates:
    cor
0.40658
```

or regress one against the other:

```
e2.change41 <- lm(changeG41 ~ changeW41, data = gastricbypassW)
summary(e2.change41)$coef
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  65.0794   24.83368  2.6206 0.017331
changeW41     1.7082    0.90473  1.8881 0.075246
```

This problem can be recast using all measurement as outcomes:

```
keep.col <- c("id","weight1","weight4","glucagonAUC1","glucagonAUC4")
gastricbypassL4 <- reshape(gastricbypassW[,keep.col], direction = "long",
                           idvar = "id", varying = 2:5, timevar = "type", v.names = "value")
gastricbypassL4$type <- factor(gastricbypassL4$type, labels = keep.col[-1])
gastricbypassL4 <- gastricbypassL4[order(gastricbypassL4$id),]
head(gastricbypassL4)
```

```
    id         type   value
1.1  1      weight1 127.200
1.2  1      weight4 108.100
1.3  1 glucagonAUC1  20.690
1.4  1 glucagonAUC4  43.434
2.1  2      weight1 165.200
2.2  2      weight4 132.000
```

fitting an unstructured mixed model:

```
e.lmm4 <- lmm(value ~ type,
              repetition = ~type|id, structure = "UN",
              data = gastricbypassL4)
```

extract the residual covariance matrix:

```
sigma.lmm4 <- sigma(e.lmm4)
sigma.lmm4
```

```
               weight1 weight4 glucagonAUC1 glucagonAUC4
weight1       410.8475  326.84       1.7077     -217.399
weight4       326.8357  290.84     -24.6003     -161.696
glucagonAUC1    1.7077  -24.60     241.7007      -81.649
glucagonAUC4 -217.3994 -161.70     -81.6493      442.464
```

Deduce the residual covariance matrix for the change:

```
Mcon <- cbind(c(-1,1,0,0),c(0,0,-1,1))
sigmeChange.lmm4 <- t(Mcon) %*% sigma.lmm4 %*% Mcon
dimnames(sigmeChange.lmm4) <- list(c("d.weight","d.glucagonAUC"),
                                   c("d.weight","d.glucagonAUC"))
sigmeChange.lmm4
```

```
              d.weight d.glucagonAUC
d.weight        48.011        82.011
d.glucagonAUC   82.011       847.464
```

and the corrrelation or covariance:

```
cov2cor(sigmeChange.lmm4)[1,2]
sigmeChange.lmm4[1,2]/sigmeChange.lmm4[1,1]
```

```
[1] 0.40658
[1] 1.7082
```

The uncertainty can be quantified using a delta method:

```
estimate(e.lmm4, function(p){
  Sigma.change <- t(Mcon) %*% sigma(e.lmm4, p = p) %*% Mcon
  c(cor = cov2cor(Sigma.change)[1,2],
    beta = Sigma.change[1,2]/Sigma.change[1,1])
})
```

```
     estimate      se     df    lower  upper p.value
cor   0.40658 0.19150 2.5925 -0.26078 1.0739 0.13791
beta  1.70818 0.88073 2.6876 -1.28836 4.7047 0.15837
```

The standard errors and degrees of freedom do not match the univariate analysis, suggesting poor small sample properties of this technic.

3 Equivalence with other R packages

3.1 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(weight ~ visit*group, structure = "RE",
               data = gastricbypassL, repetition = ~visit|id)
eCS.gls <- gls(weight ~ visit*group, correlation = corCompSymm(form=~visit|id),
               data = gastricbypassL, na.action = na.omit)
eCS.lme <- lme(weight ~ visit*group, random = ~1|id,
               data = gastricbypassL, na.action = na.omit)
logLik(eRI.lmm)
logLik(eCS.lme)
logLik(eCS.gls)
```

```
[1] -236.21
'log Lik.' -236.21 (df=10)
'log Lik.' -236.21 (df=10)
```

The estimated random effect also match:

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

```
[1] -1.7303e-08  2.6979e-08
```

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

```
eUN.gls <- gls(glucagonAUC ~ visit*group,
               correlation = corSymm(form=~as.numeric(visit)|id),
               weights = varIdent(form=~1|visit),
               data = gastricbypassL, na.action = na.omit)
logLik(eUN.gls)
logLik(eUN.lmm)
```

```
'log Lik.' -295.31 (df=18)
[1] -295.31
```

3.2 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(weight ~ visit*group + (1|id),
                 data = gastricbypassL)
logLik(eRI.lmer)
logLik(eRI.lmm)
```

```
'log Lik.' -236.21 (df=10)
[1] -236.21
```

The estimated random effects match:

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

```
[1] -1.5513e-08  2.4171e-08
```

Nested random effects correspond to block unstructured:

```
eNRI.lmm <- lmm(weight ~ visit*group, structure = RE(~(1|id/baseline)),
               data = gastricbypassL, repetition = ~visit|id)
eNRI.lmer <- lmer(weight ~ visit*group + (1|id/baseline),
                  data = gastricbypassL)
logLik(eNRI.lmer)
logLik(eNRI.lmm)
```

```
'log Lik.' -234.97 (df=11)
[1] -234.97
```

And the estimated random effects still match:

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

```
[1] -5.8317e-06  9.0913e-06
[1] -8.5850e-05  7.8971e-05
```

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

```
eUN.lmer <- lmer(glucagonAUC ~ visit*group + (0 + visit|id),
                 data = gastricbypassL,
                 control = lmerControl(check.nobs.vs.nRE = "ignore"))
logLik(eUN.lmer)
logLik(eUN.lmm)
```

```
Warning message:
In checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv,  :
  Model failed to converge with max|grad| = 0.00203036 (tol = 0.002, component 1)
'log Lik.' -295.31 (df=19)
[1] -295.31
```

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

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

```
                Multivariate Wald test

                  F-statistic       df p.value
mean: visit             5.803 (3,16.9) 0.00647 **
    : group             3.926 (1,18.0) 0.06302  .
    : visit:group       2.762 (3,17.3) 0.07332  .
```

is very similar but not identical to:

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

```
Type III Analysis of Variance Table with Satterthwaite's method
            Sum Sq Mean Sq NumDF DenDF F value  Pr(>F)
visit         1339     446     3  17.4   18.29 1.3e-05 ***
group            5       5     1  18.1    0.22   0.647
visit:group    203      68     3  17.4    2.77   0.073 .
---
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 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] -4.3812e-07  6.0172e-07
```

3.3 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
)
```

It leads nearly identical results compared to `lmm`:

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

```
Warning message:
In .lmmNormalizeData(as.data.frame(data)[unique(stats::na.omit(var.all))],  :
    3 clusters have been removed.
```

```
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] -0.00018301  0.00016268
[1] -0.00039710  0.00020479
```

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). This being said, the latest version of the package (0.3.7) included several additional extractor of model feature so this may be improved in the future.

3.4 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
```

3.5 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.

3.6 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
```

3.7 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

Vickers, A. J. and Altman, D. G. (2001). Analysing controlled trials with baseline and follow up measurements. *Bmj*, 323(7321):1123–1124.