Overview of the package LMMstar

Brice Ozenne

October 16, 2021

This vignette describes the main functionalities of the **LMMstar** package. This package implements specific types of linear mixed models mainly useful when having repeated observations over a discrete variable (e.g. time, brain region, ...). Key assumptions are that at the cluster level, observation are independent and identically distributed and that the mean and variance are driven by independent factors. In particular, in large samples the residuals do not have to be normally distributed.

The **LMMstar** package contains four main functions:

- the function `lmm` is the main function of the package which fits linear mixed models. The user can interact with *lmm* objects using:
  - `anova` to test combinations of coefficients (Wald test or Likelihood ratio tests).
  - `autoplot` to obtain a graphical display of the fitted values.
  - `coef` to extract the estimates.
  - `confint` to extract estimates, confidence intervals, and p.values.
  - `getVarCov` to extract the modeled residual variance covariance matrix.
  - `logLik` to output the log-likelihood of the estimated model.
  - `predict` to compute the conditional mean for new observations.
  - `residuals` to extract the observed residuals of the fitted model.
  - `summary` to obtain a summary of the results.
- the `summarize` function to compute summary statistics stratified on a categorical variable (typically time).
- the `sampleRem` function to simulate longitudinal data.
- the `LMMstar.options` function enables the user to display the default values used in the **LMMstar** package. The function can also change the default values to better match the user needs.

Before going further we need to load the **LMMstar** package in the R session:

```
library(LMMstar)
```

To illustrate the functionalities of the package, we will use the gastricbypass dataset:

```
data(gastricbypassL, package = "LMMstar")
head(gastricbypassL)
```

```
  id visit                    time weight glucagon
1  1     1 3 months before surgery  127.2  5032.50
2  2     1 3 months before surgery  165.2 12142.50
3  3     1 3 months before surgery  109.7 10321.35
4  4     1 3 months before surgery  146.2  6693.00
5  5     1 3 months before surgery  113.1  7090.50
6  6     1 3 months before surgery  158.8 10386.00
```

See `?gastricbypassL` for a presentation of the database. We will use a shorter version of the time variable:

```
gastricbypassL$time <- factor(gastricbypassL$time,
         levels = c("3 months before surgery", "1 week before surgery",
           "1 week after surgery", "3 months after surgery" ),
         labels = c("B3_months","B1_week","A1_week","A3_months"))
```

and rescale the glucagon values

```
gastricbypassL$glucagon <- as.double(scale(gastricbypassL$glucagon))
```

Note: the **LMMstar** package is under active development. Newer package versions may include additional functionalities and fix previous bugs. The version of the package that is being used is:

```
utils::packageVersion("LMMstar")
```

```
[1] '0.3.1'
```

When estimating model coefficients, we will use the internal optimization routine of the **LMMstar** package (instead of relying on the `nlme::gls` function, which is the default option):

```
LMMstar.options(optimizer = "FS")
```

1 Descriptive statistics

Mean, standard deviation, and other summary statistic can be computed with respect to a categorical variable (typically time) using the `summarize` function:

```
sss <- summarize(weight+glucagon ~ time, data = gastricbypassL, na.rm = TRUE)
print(sss, digits = 3)
```

```
   outcome      time observed missing     mean     sd     min   median     max
1   weight B3_months       20       0 128.9700 20.269 100.900 123.1000 173.000
2   weight   B1_week       20       0 121.2400 18.910  95.700 114.5000 162.200
3   weight   A1_week       20       0 115.7000 18.275  89.900 110.6000 155.000
4   weight A3_months       20       0 102.3650 17.054  78.800  98.5000 148.000
5 glucagon B3_months       20       0  -0.4856  0.641  -1.395  -0.6679   1.030
6 glucagon   B1_week       19       1  -0.6064  0.558  -1.416  -0.7669   0.946
7 glucagon   A1_week       19       1   1.0569  1.044  -0.478   0.9408   3.267
8 glucagon A3_months       20       0   0.0576  0.760  -1.047   0.0319   2.124
```

2 Linear mixed model

2.1 Modeling tools

Fit a linear model with **identity** structure:

```
eId.lmm <- lmm(weight ~ time + glucagon,
        repetition = ~time|id, structure = "ID",
        data = gastricbypassL)
eId.lmm
cat(" covariance structure: \n");getVarCov(eId.lmm)
```

```
     Linear regression

 outcome/cluster/time: weight/id/time
 data                : 78 observations and distributed in 20 clusters
 parameters          : 5 mean ((Intercept) timeB1_week timeA1_week timeA3_months glucagon)
                       1 variance (sigma)
 log-likelihood      : -323.086426918519
 convergence         : TRUE (6 iterations)
 covariance structure:
          B3_months  B1_week  A1_week A3_months
B3_months  330.0426   0.0000   0.0000    0.0000
B1_week      0.0000 330.0426   0.0000    0.0000
A1_week      0.0000   0.0000 330.0426    0.0000
A3_months    0.0000   0.0000   0.0000  330.0426
```

Fit a linear model with **independence** structure:

```
eInd.lmm <- lmm(weight ~ time + glucagon,
        repetition = ~time|id, structure = "IND",
        data = gastricbypassL)
eInd.lmm
cat(" covariance structure: \n");getVarCov(eInd.lmm)
```

```
     Linear regression with heterogeneous residual variance

 outcome/cluster/time: weight/id/time
 data                : 78 observations and distributed in 20 clusters
 parameters          : 5 mean ((Intercept) timeB1_week timeA1_week timeA3_months glucagon)
                       4 variance (sigma k.B1_week k.A1_week k.A3_months)
 log-likelihood      : -321.457830361849
 convergence         : TRUE (9 iterations)
 covariance structure:
          B3_months  B1_week  A1_week A3_months
B3_months  442.6475   0.0000   0.0000    0.0000
B1_week      0.0000 418.9934   0.0000    0.0000
A1_week      0.0000   0.0000 222.8463    0.0000
A3_months    0.0000   0.0000   0.0000  237.2049
```

Fit a linear mixed model with **compound symmetry** structure:

```
eCS.lmm <- lmm(weight ~ time + glucagon,
        repetition = ~time|id, structure = "CS",
        data = gastricbypassL)
eCS.lmm
cat(" covariance structure: \n");getVarCov(eCS.lmm)
```

```
     Linear Mixed Model with a compound symmetry covariance matrix

 outcome/cluster/time: weight/id/time
 data                : 78 observations and distributed in 20 clusters
 parameters          : 5 mean ((Intercept) timeB1_week timeA1_week timeA3_months glucagon)
                       1 variance (sigma)
                       1 correlation (rho)
 log-likelihood      : -243.600523870253
 convergence         : TRUE (10 iterations)
 covariance structure:
          B3_months  B1_week  A1_week A3_months
B3_months  355.3062 344.6236 344.6236  344.6236
B1_week    344.6236 355.3062 344.6236  344.6236
A1_week    344.6236 344.6236 355.3062  344.6236
A3_months  344.6236 344.6236 344.6236  355.3062
```

Fit a linear mixed model with **unstructured** covariance matrix:

```
eUN.lmm <- lmm(weight ~ time + glucagon,
        repetition = ~time|id, structure = "UN",
        data = gastricbypassL)
eUN.lmm
cat(" covariance structure: \n");getVarCov(eUN.lmm)
```

```
     Linear Mixed Model with an unstructured covariance matrix

 outcome/cluster/time: weight/id/time
 data                : 78 observations and distributed in 20 clusters
 parameters          : 5 mean ((Intercept) timeB1_week timeA1_week timeA3_months glucagon)
                       4 variance (sigma k.B1_week k.A1_week k.A3_months)
                       6 correlation (rho(B3_months,B1_week) rho(B3_months,A1_week) rho(B3_months,A
 log-likelihood      : -216.318937004305
 convergence         : TRUE (27 iterations)
 covariance structure:
          B3_months  B1_week  A1_week A3_months
B3_months  411.3114 381.9734 352.6400  318.8573
B1_week    381.9734 362.7326 335.4649  304.6314
A1_week    352.6400 335.4649 311.6921  285.8077
A3_months  318.8573 304.6314 285.8077  280.9323
```

Fit a linear mixed model with **stratified unstructured** covariance matrix:

```
gastricbypassL$group <- as.numeric(gastricbypassL$id)%%2
eSUN.lmm <- lmm(weight ~ time*group,
  repetition = group~time|id, structure = "UN",
  data = gastricbypassL)
eSUN.lmm
cat(" covariance structure: \n");getVarCov(eSUN.lmm)
```

```
     Linear Mixed Model with an unstructured covariance matrix

 outcome/cluster/time: weight/id/time
 data                : 80 observations and distributed in 20 clusters
 parameters          : 8 mean ((Intercept) timeB1_week timeA1_week timeA3_months group1 timeB1_week
                       8 variance (sigma:0 sigma:1 k.B1_week:0 k.A1_week:0 k.A3_months:0 k.B1_week:
                       12 correlation (rho(B3_months,B1_week):0 rho(B3_months,A1_week):0 rho(B3_mon
 log-likelihood      : -205.26832084513
 convergence         : TRUE (15 iterations)
 covariance structure:
$`0`
          B3_months  B1_week  A1_week A3_months
B3_months  421.2046 384.4930 373.1531  308.0198
B1_week    384.4930 363.6010 353.4851  296.0184
A1_week    373.1531 353.4851 346.9516  293.2727
A3_months  308.0198 296.0184 293.2727  260.5560

$`1`
          B3_months  B1_week  A1_week A3_months
B3_months  383.7179 360.4274 345.6647  354.9368
B1_week    360.4274 341.1832 326.9782  332.8130
A1_week    345.6647 326.9782 313.9293  319.7058
A3_months  354.9368 332.8130 319.7058  341.7246
```

2.2 Model output

The `summary` method can be used to display the main information relative to the model fit:

```
summary(eCS.lmm)
```

```
          Linear Mixed Model

Dataset: gastricbypassL

  - 20 clusters
  - 78 observations were analyzed, 2 were excluded because of missing values
  - between 3 and 4 observations per cluster

Summary of the outcome and covariates:

    $ weight  : num  127 165 110 146 113 ...
    $ time    : Factor w/ 4 levels "B3_months","B1_week",..: 1 1 1 1 1 1 1 1 1 1 ...
    $ glucagon: num  -0.9654 0.2408 -0.0682 -0.6837 -0.6163 ...
    reference level: time=B3_months

Estimation procedure

  - Restricted Maximum Likelihood (REML)
  - log-likelihood :-243.6005
  - parameters: mean = 5, variance = 1, correlation = 1
  - convergence: TRUE (10 iterations, largest |score|=3.641667e-06 is for rho)

Residual variance-covariance: compound symmetry

  - correlation structure: ~1
              B3_months B1_week A1_week A3_months
    B3_months      1.00    0.97    0.97      0.97
    B1_week        0.97    1.00    0.97      0.97
    A1_week        0.97    0.97    1.00      0.97
    A3_months      0.97    0.97    0.97      1.00

  - variance structure: ~1
          standard.deviation
    sigma           18.84957

Fixed effects: weight ~ time + glucagon

              estimate    se     df   lower   upper p.value
(Intercept)    129.369 4.226 20.034 120.556 138.183  <0.001 ***
timeB1_week     -7.619 1.054 53.968  -9.732  -5.507  <0.001 ***
timeA1_week    -14.495 1.428 53.879 -17.358 -11.632  <0.001 ***
timeA3_months  -27.051 1.087 53.943 -29.231 -24.872  <0.001 ***
```

```
glucagon          0.822  0.62  53.81  -0.421   2.065   0.191

Uncertainty was quantified using model-based standard errors (column se).
Degrees of freedom were computed using a Satterthwaite approximation (column df).
The columns lower and upper indicate a 95% confidence interval for each coefficient.
```

Note: the calculation of the degrees of freedom, especially when using the observed information can be quite slow. Setting the arguments `df` to `FALSE` and `type.information` to `"expected"` when calling `lmm` should lead to a more reasonnable computation time.

2.3 Extract estimated coefficients

The value of the estimated coefficients can be output using `coef`:

```
coef(eCS.lmm)
```

```
(Intercept)   timeB1_week   timeA1_week timeA3_months      glucagon 
129.3690995    -7.6194918   -14.4951323   -27.0514694     0.8217879 
```

It is possible to apply specific transformation on the variance coefficients, for instance to obtain the residual variance relative to each outcome:

```
coef(eUN.lmm, effects = "variance", transform.k = "sd")
```

```
sigma:B3_months    sigma:B1_week    sigma:A1_week sigma:A3_months 
       20.28081         19.04554         17.65480        16.76104 
```

2.4 Extract estimated residual variance-covariance structure

The method `getVarCov` can be used to output the covariance structure of the residuals:

```
getVarCov(eCS.lmm)
```

```
          B3_months  B1_week  A1_week A3_months
B3_months  355.3062 344.6236 344.6236  344.6236
B1_week    344.6236 355.3062 344.6236  344.6236
A1_week    344.6236 344.6236 355.3062  344.6236
A3_months  344.6236 344.6236 344.6236  355.3062
```

It can also be specific to an individual:

```
getVarCov(eCS.lmm, individual = 5)
```

```
          B3_months  A1_week A3_months
B3_months  355.3062 344.6236  344.6236
A1_week    344.6236 355.3062  344.6236
A3_months  344.6236 344.6236  355.3062
```

2.5 Model diagnostic

The method **residuals** can also be used to extract the residulas in the wide format:

```
eCS.diagW <- residuals(eCS.lmm, type = "normalized", format = "wide")
head(eCS.diagW)
```

```
  cluster  B3_months      B1_week    A1_week  A3_months
1       1 -0.8042448 -0.709908591 -1.4242830  0.3176640
2       2  1.0863177 -0.133256793  1.1083627  1.5977042
3       3 -0.4597852 -0.612727857 -0.6060136 -0.8589524
4       4 -1.0103075  0.007471092  0.1309862  1.1428822
5       5 -0.1258773           NA -0.3819184 -0.7874832
6       6  3.5646224  2.333205013  2.8387203  0.3586263
```

or in the long format:

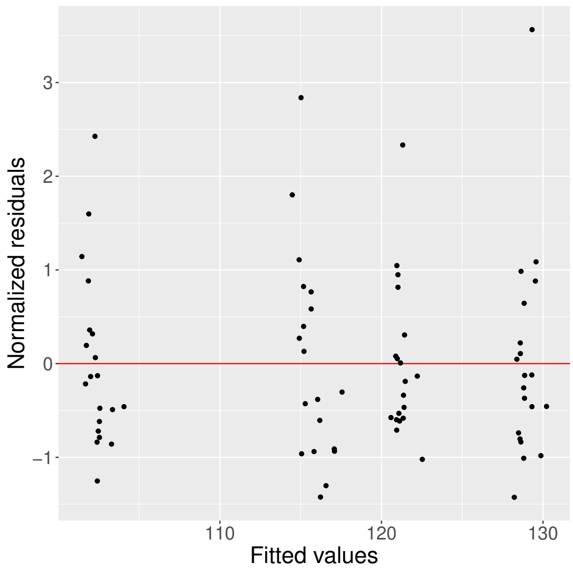
```
eCS.diagL <- residuals(eCS.lmm, type = "normalized", format = "long")
head(eCS.diagL)
```

```
[1] -0.8042448  1.0863177 -0.4597852 -1.0103075 -0.1258773  3.5646224
```

Various type of residuals can be extract but the normalized one are recommanded when doing model checking. The method **residuals** can also be used to display diagnostic plots, e.g. about:

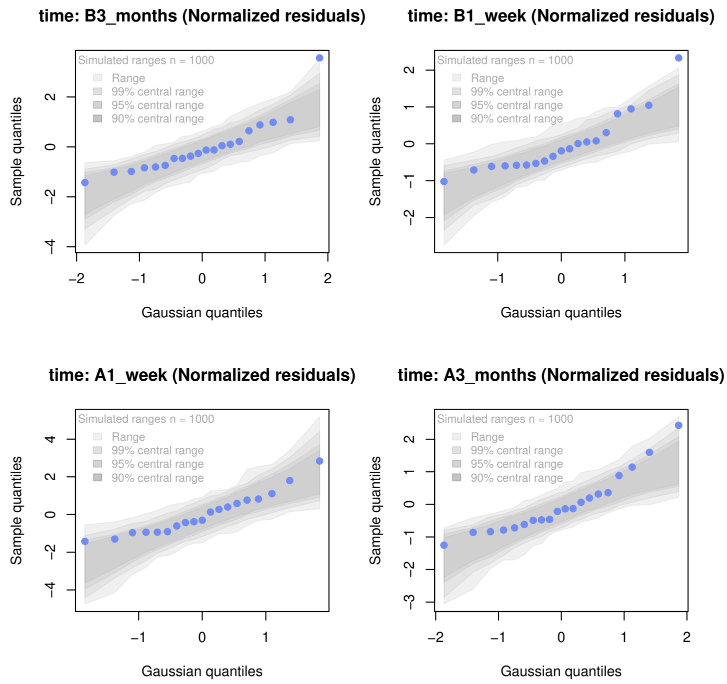
- the distribution of the residuals across fitted values using a scatterplot

```
residuals(eCS.lmm, type = "normalized", plot = "scatterplot", size.text = 20)
```



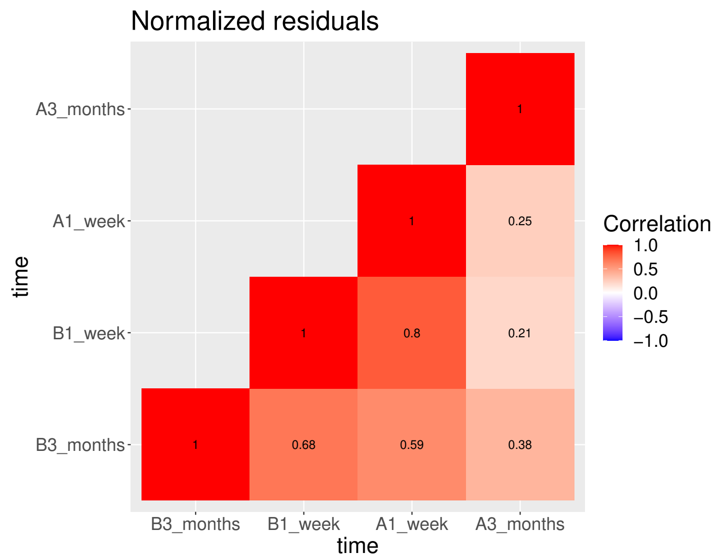
- the "normality" of the residuals at each repetition using a quantile-quantile plot [1]:

```
residuals(eCS.lmm, type = "normalized", format = "wide",
   plot = "qqplot", engine.qqplot = "qqtest")
## Note: the qqtest package to be installed to use the argument engine.plot = "qqtest"
```



- the residual correlation within cluster between the residuals:

```
residuals(eCS.lmm, type = "normalized", plot = "correlation", format = "wide",
   size.text = 20)
```



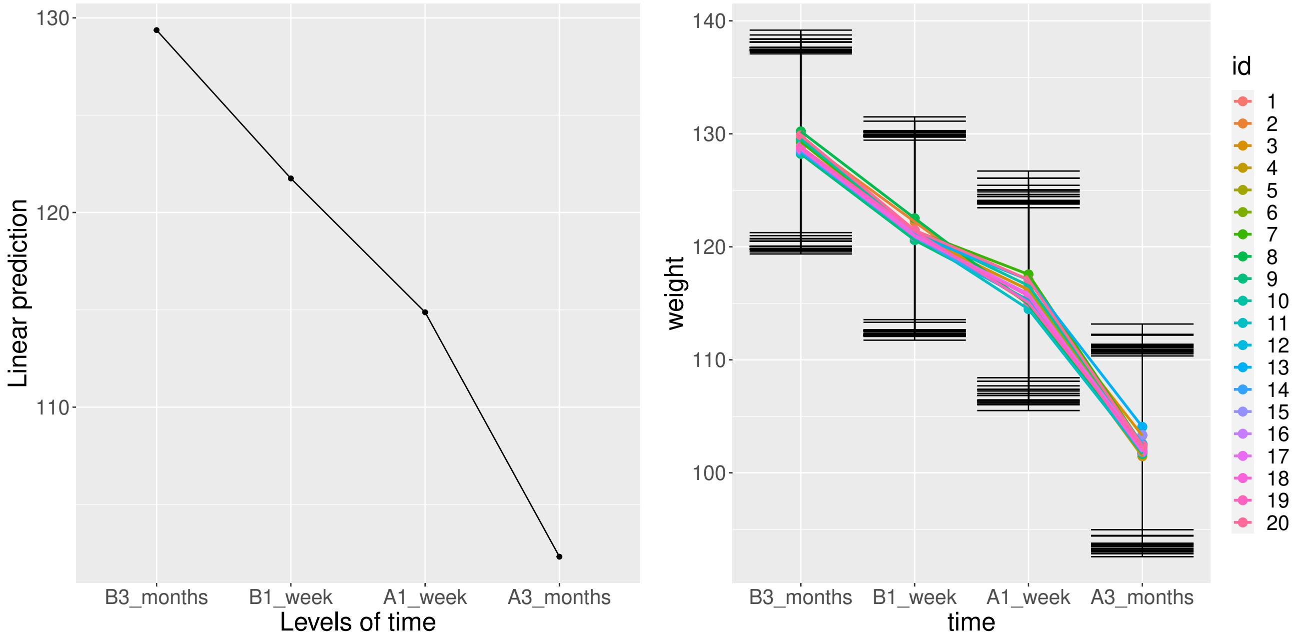
[1] see Oldford (2016) for guidance about how to read quantile-quantile plots.

2.6 Model fit

The fitted values can be displayed via the `emmeans` package or using the `autoplot` method:

```
library(emmeans) ## left panel
emmip(eCS.lmm, ~time) + theme(text = element_text(size=20))
```

```
library(ggplot2) ## right panel
autoplot(eCS.lmm, color = "id", size.text = 20)
```



In the first case the average curve (over glucago values) is displayed while in the latter each possible curve is displayed. With the `autoplot` method, it is possible to display a curve specific to a glucagon value via the argument `at`:

```
autoplot(eCS.lmm, at = data.frame(glucagon = 10), color = "glucagon")
```

2.7 Statistical inference

2.7.1 Model coefficients

The estimated coefficients with their confidence intervals can be accessed via the `confint` method:

```
confint(eCS.lmm)
```

```
              estimate   lower  upper
(Intercept)    129.369 120.556 138.18
timeB1_week     -7.619  -9.732  -5.51
timeA1_week    -14.495 -17.358 -11.63
timeA3_months  -27.051 -29.231 -24.87
glucagon         0.822  -0.421   2.06
```

Confidence intervals for the variance and correlation parameters can be displayed too specifying `effect="all"`:

```
confint(eCS.lmm, effect = "all", backtransform = TRUE)
```

```
               estimate   lower   upper
(Intercept)     129.369 120.556 138.183
timeB1_week      -7.619  -9.732  -5.507
timeA1_week     -14.495 -17.358 -11.632
timeA3_months   -27.051 -29.231 -24.872
glucagon          0.822  -0.421   2.065
sigma            18.850  13.479  26.359
rho               0.970   0.936   0.986
Note: estimates and confidence intervals for sigma, rho have been back-transformed.
```

Because these parameters are constrained (e.g. strictly positive), they uncertainty is by default computed after transformation (e.g. `log`) and then backtransformed.

2.7.2 Linear combination of the model coefficients

The `anova` method can be use to test one or several linear combinations of the model coefficients using Wald tests. For instance whether there is a change in average weight just after taking the treatment:

```
anova(eUN.lmm, effects = c("timeA1_week-timeB1_week=0"), ci = TRUE)
```

```
                    ** User-specified hypotheses **
 - F-test
 statistic df.num df.denom      p.value
  43.14135      1 17.87455 3.723358e-06

 - P-values and confidence interval
                           estimate     lower     upper      p.value
timeA1_week - timeB1_week -3.905721 -5.155643 -2.655799 3.723358e-06
```

When testing transformed variance or correlation parameters, parentheses (as in `log(k).B1_week`) cause problem for recognizing parameters:

```
try(
  anova(eUN.lmm,
 effects = c("log(k).B1_week=0","log(k).A1_week=0","log(k).A3_months=0"))
)
```

```
Error in .anova_Wald(object, effects = effects, rhs = rhs, df = df, ci = ci,  :
  Possible mispecification of the argument 'effects' as running mulcomp::glht lead to the following
Error in parse(text = ex[i]) : <text>:1:7: uventet symbol
1: log(k).B1_week
          ^
```

It is then advised to build a contrast matrix, e.g.:

```
name.coef <- rownames(confint(eUN.lmm, effects = "all", backtransform = FALSE))
name.varcoef <- grep("log(k)",name.coef, value = TRUE, fixed = TRUE)
C <- matrix(0, nrow = 3, ncol = length(name.coef), dimnames = list(name.varcoef, name.coef))
diag(C[name.varcoef,name.varcoef]) <- 1
C
```

```
                 (Intercept) timeB1_week timeA1_week timeA3_months glucagon log(sigma)
log(k).B1_week             0           0           0             0        0          0
log(k).A1_week             0           0           0             0        0          0
log(k).A3_months           0           0           0             0        0          0
                 log(k).B1_week log(k).A1_week log(k).A3_months atanh(rho(B3_months,B1_week))
log(k).B1_week                1              0                0                             0
log(k).A1_week                0              1                0                             0
log(k).A3_months              0              0                1                             0
                 atanh(rho(B3_months,A1_week)) atanh(rho(B3_months,A3_months))
log(k).B1_week                               0                               0
log(k).A1_week                               0                               0
log(k).A3_months                             0                               0
                 atanh(rho(B1_week,A1_week)) atanh(rho(B1_week,A3_months))
log(k).B1_week                             0                             0
log(k).A1_week                             0                             0
log(k).A3_months                           0                             0
                 atanh(rho(A1_week,A3_months))
log(k).B1_week                               0
log(k).A1_week                               0
log(k).A3_months                             0
```

And then call the **anova** method specifying the null hypothesis via the contrast matrix:

```
anova(eUN.lmm, effects = C)
```

```
                    ** User-specified hypotheses **
- F-test
statistic df.num df.denom     p.value
 6.203161      3 17.99456 0.004417117
```

2.8 Baseline adjustment

The `lmm` contains an "experimental" feature to drop non-identifiable effects from the model. For instance, let us define two (artifical) groups of patients:

```
gastricbypassL$group <- c("1","2")[as.numeric(gastricbypassL$id) %in% 15:20 + 1]
```

We would like to model group differences only after baseline (i.e. only at 1 week and 3 months after). For this we will define a treatment variable being the group variable except before baseline where it is "none":

```
gastricbypassL$treat <- baselineAdjustment(gastricbypassL, variable = "group",
        repetition = ~time|id, constrain = c("B3_months","B1_week"),
        new.level = "none")
table(treat = gastricbypassL$treat, time = gastricbypassL$time, group = gastricbypassL$group)
```

```
, , group = 1

      time
treat  B3_months B1_week A1_week A3_months
  none        14      14       0         0
  1            0       0      14        14
  2            0       0       0         0

, , group = 2

      time
treat  B3_months B1_week A1_week A3_months
  none         6       6       0         0
  1            0       0       0         0
  2            0       0       6         6
```

Here we will be able to estimate a total of 6 means and therefore can at most identify 6 effects. However the design matrix for the interaction model:

```
colnames(model.matrix(weight ~ treat*time, data = gastricbypassL))
```

```
[1] "(Intercept)"          "treat1"               "treat2"               "timeB1_week"
[5] "timeA1_week"          "timeA3_months"        "treat1:timeB1_week"   "treat2:timeB1_week"
[9] "treat1:timeA1_week"   "treat2:timeA1_week"   "treat1:timeA3_months" "treat2:timeA3_months"
```

contains 12 parameters (i.e. 6 too many). The `lmm` function will internally remove the one that cannot be identified and fit a simplified model:

```
eC.lmm <- lmm(weight ~ treat*time, data = gastricbypassL,
       repetition = ~time|id, structure = "UN")
```

```
Advarselsbesked:
I .model.matrix_regularize(formula, data) :
  Constant values in the design matrix in interactions "treat:time"
 Coefficients "treat1" "treat2" "timeA1_week" "timeA3_months" "treat1:timeB1_week" "treat2:timeB1_w
Consider defining manually the interaction, e.g. via droplevels(interaction(.,.)) to avoid this war
```

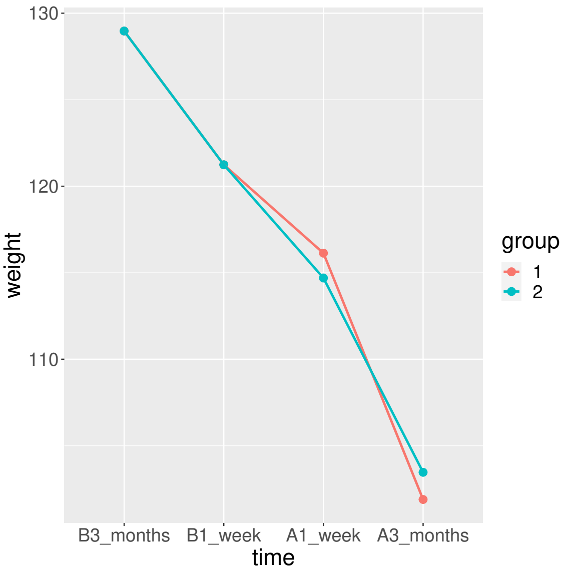
with the following coefficients:

```
coef(eC.lmm, effects = "mean")
```

```
        (Intercept)          timeB1_week   treat1:timeA1_week   treat2:timeA1_week
          128.97000             -7.73000            -12.83949            -14.27452
treat1:timeA3_months treat2:timeA3_months
          -27.07620            -25.50553
```

One can vizualize the baseline adjustment via the `autoplot` function:

```
autoplot(eC.lmm, color = "group", ci = FALSE, size.text = 20)
```



To more easily compare the two groups, one could set the baseline treatment to the treatment in the control arm by omitting the argument `new.level`:

```
gastricbypassL$treat2 <- baselineAdjustment(gastricbypassL, variable = "group",
         repetition = ~time|id, constrain = c("B3_months","B1_week"))
table(treat = gastricbypassL$treat2, time = gastricbypassL$time, group = gastricbypassL$group)
```

```
windows
      2
, , group = 1

     time
treat B3_months B1_week A1_week A3_months
    1        14      14      14        14
    2         0       0       0         0

, , group = 2

     time
treat B3_months B1_week A1_week A3_months
    1         6       6       0         0
    2         0       0       6         6
```

Fitting the model

```
eC2.lmm <- suppressWarnings(lmm(weight ~ treat2*time, data = gastricbypassL,
    repetition = ~time|id, structure = "UN"))
```

will directly output group differences (last two coefficients):

```
confint(eC2.lmm, effects = "mean", columns = c("estimate","lower","upper","p.value"))
```

```
                     estimate  lower  upper  p.value
(Intercept)            128.97 119.48 138.46 0.00e+00
timeB1_week             -7.73  -9.19  -6.27 1.00e-09
timeA1_week            -12.84 -14.64 -11.04 2.02e-12
timeA3_months          -27.08 -30.66 -23.50 3.20e-13
treat22:timeA1_week     -1.44  -2.75  -0.12 3.43e-02
treat22:timeA3_months    1.57  -3.64   6.78 5.32e-01
```

It is also possible to get the estimated mean at each timepoint, using an equivalent mean structure:

```
eC3.lmm <- suppressWarnings(lmm(weight ~ 0+treat2:time, data = gastricbypassL,
    repetition = ~time|id, structure = "UN"))
confint(eC3.lmm)
```

```
                     estimate lower upper
treat21:timeB3_months      129 119.5   138
treat21:timeB1_week        121 112.4   130
treat21:timeA1_week        116 107.5   125
treat22:timeA1_week        115 106.1   123
treat21:timeA3_months      102  93.8   110
treat22:timeA3_months      103  94.9   112
```

or the baseline mean and the change since baseline:

```
eC4.lmm <- suppressWarnings(lmm(weight ~ treat2:time, data = gastricbypassL,
    repetition = ~time|id, structure = "UN"))
confint(eC4.lmm)
```

```
                     estimate  lower  upper
(Intercept)            128.97 119.48 138.46
treat21:timeB1_week     -7.73  -9.19  -6.27
treat21:timeA1_week    -12.84 -14.64 -11.04
treat22:timeA1_week    -14.27 -16.23 -12.32
treat21:timeA3_months  -27.08 -30.66 -23.50
treat22:timeA3_months  -25.51 -30.32 -20.69
```

2.9 Marginal means

The `lmm` function can be used in conjonction with the `emmeans` package to compute marginal means. Consider the following model:

```
e.group <- lmm(weight ~ time*group, data = gastricbypassL,
        repetition = ~time|id, structure = "UN")
```

We can for instance compute the average value over time *assuming balanced groups*:

```
library(emmeans)
emmeans(e.group, specs=~time)
```

```
NOTE: Results may be misleading due to involvement in interactions
 time       emmean   SE   df lower.CL upper.CL
 B3_months     130 5.05 18.0    119.3      141
 B1_week       122 4.69 18.0    112.5      132
 A1_week       117 4.55 18.0    107.0      126
 A3_months     104 4.20 18.1     94.9      113

Results are averaged over the levels of: group
Confidence level used: 0.95
```

This differs from the average value over time over the whole sample:

```
df.pred <- cbind(gastricbypassL, predict(e.group, newdata = gastricbypassL))
summarize(formula = estimate~time, data = df.pred)
```

```
   outcome      time observed missing    mean       sd      min   median    max
1 estimate B3_months       20       0 128.970 2.270212 127.5214 127.5214 132.35
2 estimate   B1_week       20       0 121.240 2.726942 119.5000 119.5000 125.30
3 estimate   A1_week       20       0 115.700 2.014981 114.4143 114.4143 118.70
4 estimate A3_months       20       0 102.365 3.146729 100.3571 100.3571 107.05
```

as the groups are not balanced:

```
table(group = gastricbypassL$group, time = gastricbypassL$time)
```

```
     time
group B3_months B1_week A1_week A3_months
    1        14      14      14        14
    2         6       6       6         6
```

The "emmeans" approach gives equal "weight" to the expected value of both group 2 (instead of less weight for group 2). By hand:

```
mu.group1 <-  as.double(coef(e.group)["(Intercept)"])
mu.group2 <-  as.double(coef(e.group)["(Intercept)"] + coef(e.group)["group2"])
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
129.9357 128.9700
```

Which one is relevant depends on the application. The `emmeans` function can also be used to display expected value in each group over time:

```
emmeans.group <- emmeans(e.group, specs = ~group|time)
emmeans.group
```

```
time = B3_months:
 group emmean   SE   df lower.CL upper.CL
 1        128 5.53 18.0    115.9      139
 2        132 8.45 18.0    114.6      150

time = B1_week:
 group emmean   SE   df lower.CL upper.CL
 1        120 5.14 18.0    108.7      130
 2        125 7.85 18.0    108.8      142

time = A1_week:
 group emmean   SE   df lower.CL upper.CL
 1        114 4.99 18.0    103.9      125
 2        119 7.62 18.0    102.7      135

time = A3_months:
 group emmean   SE   df lower.CL upper.CL
 1        100 4.60 18.1     90.7      110
 2        107 7.03 18.1     92.3      122

Confidence level used: 0.95
```

Using the pair function displays the differences:

```
epairs.group <- pairs(emmeans.group, reverse = TRUE)
epairs.group
```

```
time = B3_months:
 contrast estimate    SE   df t.ratio p.value
 2 - 1        4.83 10.10 18.0   0.478  0.6383

time = B1_week:
 contrast estimate    SE   df t.ratio p.value
 2 - 1        5.80  9.38 18.0   0.618  0.5441

time = A1_week:
 contrast estimate    SE   df t.ratio p.value
 2 - 1        4.29  9.11 18.0   0.471  0.6435

time = A3_months:
 contrast estimate    SE   df t.ratio p.value
 2 - 1        6.69  8.40 18.1   0.797  0.4361
```

One can adjust for multiple comparison via the `adjust` argument and display confidence intervals setting the argument `infer` to TRUE:

```
summary(epairs.group, by = NULL, adjust = "mvt", infer = TRUE)
```

```
 contrast time      estimate    SE   df lower.CL upper.CL t.ratio p.value
 2 - 1    B3_months     4.83 10.10 18.0    -18.0     27.7   0.478  0.7498
 2 - 1    B1_week       5.80  9.38 18.0    -15.4     27.0   0.618  0.6488
 2 - 1    A1_week       4.29  9.11 18.0    -16.3     24.9   0.471  0.7552
 2 - 1    A3_months     6.69  8.40 18.1    -12.3     25.7   0.797  0.5284

Confidence level used: 0.95
Conf-level adjustment: mvt method for 4 estimates
P value adjustment: mvt method for 4 tests
```

This should also work when doing baseline adjustment (because of baseline adjustment no difference is expected at the first two timepoints):

```
summary(pairs(emmeans(eC2.lmm , specs = ~treat2|time), reverse = TRUE), by = NULL)
```

```
Note: adjust = "tukey" was changed to "sidak"
because "tukey" is only appropriate for one set of pairwise comparisons
 contrast time      estimate    SE   df t.ratio p.value
 2 - 1    B3_months     0.00 0.000  NaN     NaN     NaN
 2 - 1    B1_week       0.00 0.000  NaN     NaN     NaN
 2 - 1    A1_week      -1.44 0.621 16.2  -2.311  0.1303
 2 - 1    A3_months     1.57 2.463 16.3   0.638  0.9522

P value adjustment: sidak method for 4 tests
```

2.10 Predictions

Two types of predictions can be performed with the `predict` method:

- **static predictions** that are only conditional on the covariates:

```
news <- gastricbypassL[gastricbypassL$id==1,]
news$glucagon <- 0
predict(eCS.lmm, newdata = news)
```

```
  estimate       se       df     lower    upper
1 129.3691 4.225632 20.03432 120.55555 138.1826
2 121.7496 4.235605 20.22155 112.92049 130.5787
3 114.8740 4.271415 20.89949 105.98847 123.7595
4 102.3176 4.215043 19.83701  93.52057 111.1147
```

which can be computing by creating a design matrix:

```
X.12 <- model.matrix(formula(eCS.lmm), news)
X.12
```

```
   (Intercept) timeB1_week timeA1_week timeA3_months glucagon
1            1           0           0             0        0
21           1           1           0             0        0
41           1           0           1             0        0
61           1           0           0             1        0
attr(,"assign")
[1] 0 1 1 1 2
attr(,"contrasts")
attr(,"contrasts")$time
[1] "contr.treatment"
```

and then multiplying it with the regression coefficients:

```
X.12 %*% coef(eCS.lmm)
```

```
       [,1]
1  129.3691
21 121.7496
41 114.8740
61 102.3176
```

- **dynamic predictions** that are conditional on the covariates and the outcome measured at other timepoints. Consider two subjects for who we would like to predict the weight 1 week before the intervention based on the weight 3 months before the intervention:

```
newd <- rbind(
  data.frame(id = 1, time = "B3_months", weight = coef(eCS.lmm)["(Intercept)"], glucagon = 0),
  data.frame(id = 1, time = "B1_week", weight = NA, glucagon = 0),
  data.frame(id = 2, time = "B3_months", weight = 100, glucagon = 0),
  data.frame(id = 2, time = "B1_week", weight = NA, glucagon = 0)
)
predict(eCS.lmm, newdata = newd, type = "dynamic", keep.newdata = TRUE)
```

```
  id      time   weight glucagon  estimate       se  df     lower    upper
1  1 B3_months 129.3691        0        NA       NA  NA        NA       NA
2  1   B1_week       NA        0 121.74961 1.046825 Inf 119.69787 123.8013
3  2 B3_months 100.0000        0        NA       NA  NA        NA       NA
4  2   B1_week       NA        0  93.26352 5.603475 Inf  82.28091 104.2461
```

The first subjects has the average weight while the second has a much lower weight. The predicted weight for the first subject is then the average weight one week before while it is lower for the second subject due to the positive correlation over time. The predicted value is computed using the formula of the conditional mean for a Gaussian vector:

```
mu1 <- coef(eCS.lmm)[1]
mu2 <- sum(coef(eCS.lmm)[1:2])
Omega_11 <- getVarCov(eCS.lmm)["B3_months","B3_months"]
Omega_21 <- getVarCov(eCS.lmm)["B1_week","B3_months"]
as.double(mu2 + Omega_21 * (100 - mu1) / Omega_11)
```

```
[1] 93.26352
```

3 Data generation

Simulate some data in the wide format:

```
set.seed(10) ## ensure reproductibility
n.obs <- 100
n.times <- 4
mu <- rep(0,4)
gamma <- matrix(0, nrow = n.times, ncol = 10) ## add interaction
gamma[,6] <- c(0,1,1.5,1.5)
dW <- sampleRem(n.obs, n.times = n.times, mu = mu, gamma = gamma, format = "wide")
head(round(dW,3))
```

```
  id X1 X2 X3 X4 X5     X6     X7     X8    X9    X10     Y1     Y2     Y3     Y4
1  1  1  0  1  1  0 -0.367  1.534 -1.894 1.729  0.959  1.791  2.429  3.958  2.991
2  2  1  0  1  2  0 -0.410  2.065  1.766 0.761 -0.563  2.500  4.272  3.002  2.019
3  3  0  0  2  1  0 -1.720 -0.178  2.357 1.966  1.215 -3.208 -5.908 -4.277 -5.154
4  4  0  0  0  1  0  0.923 -2.089  0.233 1.307 -0.906 -2.062  0.397  1.757 -1.380
5  5  0  0  2  1  0  0.987  5.880  0.385 0.028  0.820  7.963  7.870  7.388  8.609
6  6  0  0  1  1  2 -1.075  0.479  2.202 0.900 -0.739  0.109 -1.602 -1.496 -1.841
```

Simulate some data in the long format:

```
set.seed(10) ## ensure reproductibility
dL <- sampleRem(n.obs, n.times = n.times, mu = mu, gamma = gamma, format = "long")
head(dL)
```

```
  id visit        Y X1 X2 X3 X4 X5         X6       X7        X8        X9        X10
1  1     1 1.791444  1  0  1  1  0 -0.3665251 1.533815 -1.894425 1.7288665  0.9592499
2  1     2 2.428570  1  0  1  1  0 -0.3665251 1.533815 -1.894425 1.7288665  0.9592499
3  1     3 3.958350  1  0  1  1  0 -0.3665251 1.533815 -1.894425 1.7288665  0.9592499
4  1     4 2.991198  1  0  1  1  0 -0.3665251 1.533815 -1.894425 1.7288665  0.9592499
5  2     1 2.500179  1  0  1  2  0 -0.4097541 2.065413  1.765841 0.7613348 -0.5630173
6  2     2 4.272357  1  0  1  2  0 -0.4097541 2.065413  1.765841 0.7613348 -0.5630173
```

4 Modifying default options

The `LMMstar.options` method enable to get and set the default options used by the package. For instance, the default option for the information matrix is:

```
LMMstar.options("type.information")
```

```
$type.information
[1] "observed"
```

To change the default option to "expected" (faster to compute but less accurate p-values and confidence intervals in small samples) use:

```
LMMstar.options(type.information = "expected")
```

To restore the original default options do:

```
LMMstar.options(reinitialise = TRUE)
```

5 R session

Details of the R session used to generate this document:

```
sessionInfo()
```

```
R version 4.1.1 (2021-08-10)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 19042)

Matrix products: default

locale:
[1] LC_COLLATE=Danish_Denmark.1252  LC_CTYPE=Danish_Denmark.1252    LC_MONETARY=Danish_Denmark.1252
[4] LC_NUMERIC=C                    LC_TIME=Danish_Denmark.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] emmeans_1.6.3     LMMstar_0.3.0     nlme_3.1-152      ggplot2_3.3.5     spelling_2.2
 [6] roxygen2_7.1.1    butils.base_1.2   Rcpp_1.0.7        data.table_1.14.0 devtools_2.4.2
[11] usethis_2.0.1

loaded via a namespace (and not attached):
 [1] pkgload_1.2.1       splines_4.1.1       remotes_2.4.0       sessioninfo_1.1.1
 [5] globals_0.14.0      numDeriv_2016.8-1.1 pillar_1.6.3        lattice_0.20-44
 [9] glue_1.4.2          digest_0.6.27       colorspace_2.0-2    sandwich_3.0-1
[13] qqtest_1.2.0        plyr_1.8.6          Matrix_1.3-4        pkgconfig_2.0.3
[17] listenv_0.8.0       purrr_0.3.4         xtable_1.8-4        mvtnorm_1.1-2
[21] scales_1.1.1        processx_3.5.2      lava_1.6.10         tibble_3.1.4
[25] farver_2.1.0        generics_0.1.0      ellipsis_0.3.2      TH.data_1.1-0
[29] cachem_1.0.6        withr_2.4.2         cli_3.0.1           survival_3.2-11
[33] magrittr_2.0.1      crayon_1.4.1        memoise_2.0.0       estimability_1.3
[37] ps_1.6.0            fs_1.5.0            fansi_0.5.0         future_1.22.1
[41] parallelly_1.28.1   MASS_7.3-54         xml2_1.3.2          pkgbuild_1.2.0
[45] tools_4.1.1         prettyunits_1.1.1   lifecycle_1.0.1     multcomp_1.4-17
[49] stringr_1.4.0       munsell_0.5.0       callr_3.7.0         compiler_4.1.1
[53] rlang_0.4.11        grid_4.1.1          labeling_0.4.2      testthat_3.0.4
[57] gtable_0.3.0        codetools_0.2-18    reshape2_1.4.4      R6_2.5.1
[61] zoo_1.8-9           knitr_1.33          dplyr_1.0.7         fastmap_1.1.0
[65] future.apply_1.8.1  utf8_1.2.2          rprojroot_2.0.2     desc_1.3.0
[69] stringi_1.7.4       parallel_4.1.1      vctrs_0.3.8         tidyselect_1.1.1
[73] xfun_0.25           coda_0.19-4
```

References

Oldford, R. W. (2016). Self-calibrating quantile–quantile plots. *The American Statistician*, 70(1):74–90.

Appendix A Likelihood in a linear mixed model

A.1 Log-likelihood

Denote by $\boldsymbol{Y}$ a vector of m outcomes, $\boldsymbol{X}$ a vector of p covariates, $\mu(\boldsymbol{\Theta}, \boldsymbol{X})$ the modeled mean, and $\Omega(\boldsymbol{\Theta}, \boldsymbol{X})$ the modeled residual variance-covariance. The restricted log-likelihood in a linear mixed model can then be written:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\Theta}|\boldsymbol{Y},\boldsymbol{X}) =& \frac{p}{2}\log(2\pi) - \frac{1}{2}\log\left(\left|\sum_{i=1}^n \boldsymbol{X}_i \Omega_i^{-1}(\boldsymbol{\Theta})\boldsymbol{X}_i^\intercal\right|\right) \\ &+ \sum_{i=1}^n \left(-\frac{m}{2}\log(2\pi) - \frac{1}{2}\log|\Omega_i(\boldsymbol{\Theta})| - \frac{1}{2}(\boldsymbol{Y}_i - \mu(\boldsymbol{\Theta},\boldsymbol{X}_i))\Omega_i(\boldsymbol{\Theta})^{-1}(\boldsymbol{Y}_i - \mu(\boldsymbol{\Theta},\boldsymbol{X}_i))^\intercal\right) \quad \text{(A)}\end{aligned}$$

This is what the `logLik` method is computing for the REML criteria. The red term is specific to the REML criteria and prevents from computing individual contributions to the likelihood[2]. The blue term is what `logLik` outputs for the ML criteria when setting the argument `indiv` to `TRUE`.

A.2 Score

Using that $\partial \log(\det(X)) = tr(X^{-1}\partial(X))$, the score is obtained by derivating once the log-likelihood, i.e., for $\theta \in \boldsymbol{\Theta}$:

$$\begin{aligned}\mathcal{S}(\theta) =& \frac{\partial \mathcal{L}(\boldsymbol{\Theta}|\boldsymbol{Y},\boldsymbol{X})}{\partial\theta} = \frac{1}{2}tr\left(\left(\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\boldsymbol{\Theta})\boldsymbol{X}_i^\intercal\right)^{-1}\left(\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\boldsymbol{\Theta})\frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta}\Omega_i(\boldsymbol{\Theta})^{-1}\boldsymbol{X}_i^\intercal\right)\right) \\ &+ \sum_{i=1}^n\left(-\frac{1}{2}tr\left(\Omega_i(\boldsymbol{\Theta})^{-1}\frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta}\right) + \frac{\partial\mu(\boldsymbol{\Theta},\boldsymbol{X}_i)}{\partial\theta}\Omega_i(\boldsymbol{\Theta})^{-1}(\boldsymbol{Y}_i - \mu(\boldsymbol{\Theta},\boldsymbol{X}_i))^\intercal\right. \\ &\left. + \frac{1}{2}(\boldsymbol{Y}_i - \mu(\boldsymbol{\Theta},\boldsymbol{X}_i))\Omega_i(\boldsymbol{\Theta})^{-1}\frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta}\Omega_i(\boldsymbol{\Theta})^{-1}(\boldsymbol{Y}_i - \mu(\boldsymbol{\Theta},\boldsymbol{X}_i))^\intercal\right).\end{aligned}$$

This is what the `score` method is computing for the REML criteria. The red term is specific to the REML criteria and prevents from computing the score relative to each cluster. The blue term is what `score` outputs for the ML criteria when setting the argument `indiv` to `TRUE`.

[2] The REML is the likelihood of the observations divided by the prior on the estimated mean parameters $\widehat{\boldsymbol{\Theta}}_\mu \sim \mathcal{N}\left(\mu, \left(\boldsymbol{X}\Omega^{-1}(\boldsymbol{\Theta})\boldsymbol{X}^\intercal\right)^{-1}\right)$. This corresponds to $\frac{1}{\sqrt{2\pi}^p\left|\left(\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\boldsymbol{\Theta})\boldsymbol{X}_i^\intercal\right)^{-1}\right|}\exp\left(-(\widehat{\boldsymbol{\Theta}}_\mu - \mu)\left(2\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\boldsymbol{\Theta})\boldsymbol{X}_i^\intercal\right)^{-1}(\widehat{\boldsymbol{\Theta}}_\mu - \mu)^\intercal\right)$ Since μ will be estimated to be $\boldsymbol{\Theta}_\mu$, the exponential term equals 1 and thus does not contribute to the log-likelihood. One divided by the other term gives $\sqrt{2\pi}^p\left(\left|\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\boldsymbol{\Theta})\boldsymbol{X}_i^\intercal\right|\right)^{-1}$. The log of this term equals the red term

A.3 Hessian

Derivating a second time the log-likelihood gives the hessian, $\mathcal{H}(\mathbf{\Theta})$, with element[3]:

$$
\begin{aligned}
\mathcal{H}(\theta,\theta') &= \frac{\partial^2 \mathcal{L}(\mathbf{\Theta}|\boldsymbol{Y},\boldsymbol{X})}{\partial\theta\partial\theta'} = \frac{\partial \mathcal{S}(\theta)}{\partial\theta'} \\
=&\frac{1}{2}tr\left(\left(\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\mathbf{\Theta})\boldsymbol{X}_i^\intercal\right)^{-1}\left\{\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\mathbf{\Theta})\left(\frac{\partial^2\Omega_i(\mathbf{\Theta})}{\partial\theta\partial\theta'} - 2\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta}\Omega_i^{-1}(\mathbf{\Theta})\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta'}\right)\Omega_i(\mathbf{\Theta})^{-1}\boldsymbol{X}_i^\intercal\right.\right. \\
&\left.\left.+\left(\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\mathbf{\Theta})\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta}\Omega_i(\mathbf{\Theta})^{-1}\boldsymbol{X}_i^\intercal\right)\left(\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\mathbf{\Theta})\boldsymbol{X}_i^\intercal\right)^{-1}\left(\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\mathbf{\Theta})\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta'}\Omega_i(\mathbf{\Theta})^{-1}\boldsymbol{X}_i^\intercal\right)\right\}\right) \\
&+\sum_{i=1}^n\left(\frac{1}{2}tr\left(\Omega_i(\mathbf{\Theta})^{-1}\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta'}\Omega_i(\mathbf{\Theta})^{-1}\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta} - \Omega_i(\mathbf{\Theta})^{-1}\frac{\partial^2\Omega_i(\mathbf{\Theta})}{\partial\theta\partial\theta'}\right)\right. \\
&-\frac{\partial\mu(\mathbf{\Theta},\boldsymbol{X}_i)}{\partial\theta}\Omega_i(\mathbf{\Theta})^{-1}\frac{\partial\Omega_i(\mathbf{\Theta})^{-1}}{\partial\theta'}\Omega_i(\mathbf{\Theta})^{-1}\boldsymbol{\varepsilon}_i(\mathbf{\Theta})^\intercal - \frac{\partial\mu(\mathbf{\Theta},\boldsymbol{X}_i)}{\partial\theta}\Omega_i(\mathbf{\Theta})^{-1}\frac{\partial\mu(\mathbf{\Theta},\boldsymbol{X}_i)}{\partial\theta'}^\intercal \\
&\left.+\frac{1}{2}\boldsymbol{\varepsilon}_i(\mathbf{\Theta})\Omega_i(\mathbf{\Theta})^{-1}\left(\frac{\partial^2\Omega_i(\mathbf{\Theta})}{\partial\theta\partial\theta'} - \frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta'}\Omega_i(\mathbf{\Theta})^{-1}\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta} - \frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta}\Omega_i(\mathbf{\Theta})^{-1}\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta'}\right)\Omega_i(\mathbf{\Theta})^{-1}\boldsymbol{\varepsilon}_i(\mathbf{\Theta})^\intercal\right).
\end{aligned}
$$

where $\boldsymbol{\varepsilon}_i(\mathbf{\Theta}) = \boldsymbol{Y}_i - \mu(\mathbf{\Theta},\boldsymbol{X}_i)$.

The `information` method will (by default) return the (observed) information which is the opposite of the hessian. So multiplying the previous formula by -1 gives what `information` output for the REML criteria. The red term is specific to the REML criteria and prevents from computing the information relative to each cluster. The blue term is what `information` outputs for the ML criteria (up to a factor -1) when setting the argument `indiv` to `TRUE`.

A possible simplification is to use the expected hessian at the maximum likelihood. Indeed for any deterministic matrix A:

- $\mathbb{E}\left[A(\boldsymbol{Y}_i - \mu(\mathbf{\Theta},\boldsymbol{X}_i))^\intercal|\boldsymbol{X}_i\right] = 0$
- $\mathbb{E}\left[(\boldsymbol{Y}_i - \mu(\mathbf{\Theta},\boldsymbol{X}_i))A(\boldsymbol{Y}_i - \mu(\mathbf{\Theta},\boldsymbol{X}_i))^\intercal||\boldsymbol{X}_i\right] = tr(A\mathbb{V}ar(\boldsymbol{Y}_i - \mu(\mathbf{\Theta},\boldsymbol{X}_i)))$

when $\mathbb{E}\left[\boldsymbol{Y}_i - \mu(\mathbf{\Theta},\boldsymbol{X}_i)\right] = 0$. This leads to:

$$
\begin{aligned}
&\mathbb{E}\left[\mathcal{H}(\theta,\theta')|\boldsymbol{X}\right] \\
&= \frac{1}{2}tr\left(\left(\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\mathbf{\Theta})\boldsymbol{X}_i^\intercal\right)^{-1}\left\{\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\mathbf{\Theta})\left(\frac{\partial^2\Omega_i(\mathbf{\Theta})}{\partial\theta\partial\theta'} - 2\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta}\Omega_i^{-1}(\mathbf{\Theta})\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta'}\right)\Omega_i(\mathbf{\Theta})^{-1}\boldsymbol{X}_i^\intercal\right.\right. \\
&\quad\left.\left.+\left(\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\mathbf{\Theta})\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta}\Omega_i(\mathbf{\Theta})^{-1}\boldsymbol{X}_i^\intercal\right)\left(\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\mathbf{\Theta})\boldsymbol{X}_i^\intercal\right)^{-1}\left(\sum_{i=1}^n \boldsymbol{X}_i\Omega_i^{-1}(\mathbf{\Theta})\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta'}\Omega_i(\mathbf{\Theta})^{-1}\boldsymbol{X}_i^\intercal\right)\right\}\right) \\
&+\sum_{i=1}^n\left(-\frac{1}{2}tr\left(\Omega_i(\mathbf{\Theta})^{-1}\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta'}\Omega_i(\mathbf{\Theta})^{-1}\frac{\partial\Omega_i(\mathbf{\Theta})}{\partial\theta}\right) - \frac{\partial\mu(\mathbf{\Theta},\boldsymbol{X}_i)}{\partial\theta}\Omega_i(\mathbf{\Theta})^{-1}\frac{\partial\mu(\mathbf{\Theta},\boldsymbol{X}_i)}{\partial\theta'}^\intercal\right) \qquad \text{(B)}
\end{aligned}
$$

This is what `information` output when the argument `type.information` is set to `"expected"` (up to a factor -1).

[3] if one is relative to the mean and the other to the variance then they are respectively θ and θ'

A.4 Degrees of freedom

Degrees of freedom are computed using a Satterthwaite approximation, i.e. for an estimate coefficient $\widehat{\beta} \in \widehat{\boldsymbol{\Theta}}$ with standard error $\sigma_{\widehat{beta}}$, the degree of freedom is:

$$df\left(\sigma_{\widehat{\beta}}\right) = \frac{2\sigma_{\widehat{\beta}}}{\mathbb{V}ar\left[\widehat{\sigma}_{\widehat{\beta}}\right]}$$

Using a first order Taylor expansion we can approximate the variance term as:

$$\begin{aligned}
\mathbb{V}ar\left[\widehat{\sigma}_{\widehat{\beta}}\right] &\approx \frac{\partial \widehat{\sigma}_{\widehat{\beta}}}{\partial \boldsymbol{\Theta}} \Sigma_{\boldsymbol{\Theta}} \frac{\partial \widehat{\sigma}_{\widehat{\beta}}}{\partial \boldsymbol{\Theta}}^{\intercal} \\
&\approx c_\beta \left(\widehat{\mathcal{I}}_{\widehat{\boldsymbol{\Theta}}}\right)^{-1} \frac{\partial \widehat{\mathcal{I}}_{\widehat{\boldsymbol{\Theta}}}}{\partial \boldsymbol{\Theta}} \left(\widehat{\mathcal{I}}_{\widehat{\boldsymbol{\Theta}}}\right)^{-1} c_\beta{}^{\intercal} \Sigma_{\boldsymbol{\Theta}} c_\beta{}^{\intercal} \left(\widehat{\mathcal{I}}_{\widehat{\boldsymbol{\Theta}}}\right)^{-1} \frac{\partial \widehat{\mathcal{I}}_{\widehat{\boldsymbol{\Theta}}}}{\partial \boldsymbol{\Theta}}^{\intercal} \left(\widehat{\mathcal{I}}_{\widehat{\boldsymbol{\Theta}}}\right)^{-1} c_\beta
\end{aligned}$$

where $\Sigma_{\boldsymbol{\Theta}}$ is the variance-covariance matrix of all model coefficients, $\mathcal{I}_{\boldsymbol{\Theta}}$ the information matrix for all model coefficients, c_β a matrix used to select the element relative to β in the first derivative of the information matrix, and $\frac{\partial .}{\partial \boldsymbol{\Theta}}$ denotes the vector of derivatives with respect to all model coefficients.

The derivative of the information matrix (i.e. negative hessian) can then be computed using numerical derivatives or using analytical formula. To simplify the derivation of the formula we will only derive them at the maximum likelihood, i.e. when $\mathbb{E}\left[\frac{\partial \mathcal{H}(\theta,\theta'|\boldsymbol{X})}{\partial \theta''}\right] = \frac{\partial \mathbb{E}[\mathcal{H}(\theta,\theta'|\boldsymbol{X})]}{\partial \theta''}$ where the expectation is taken over $\boldsymbol{X}$. We can therefore take the derivative of formula (B). We first note that its derivative with respect to the mean parameters is 0. So we just need to compute the derivative with respect to a variance parameter θ'':

$$\begin{aligned}
&\frac{\partial \mathbb{E}\left[\mathcal{H}(\theta,\theta')|\boldsymbol{X}\right]}{\partial \theta''} \\
&+ \sum_{i=1}^{n} \left(-\frac{1}{2} tr \left(-2\Omega_i(\boldsymbol{\Theta})^{-1} \frac{\partial \Omega_i(\boldsymbol{\Theta})}{\partial \theta''} \Omega_i(\boldsymbol{\Theta})^{-1} \frac{\partial \Omega_i(\boldsymbol{\Theta})}{\partial \theta'} \Omega_i(\boldsymbol{\Theta})^{-1} \frac{\partial \Omega_i(\boldsymbol{\Theta})}{\partial \theta} \right.\right. \\
&\qquad \left. + \Omega_i(\boldsymbol{\Theta})^{-1} \frac{\partial^2 \Omega_i(\boldsymbol{\Theta})}{\partial \theta' \partial \theta''} \Omega_i(\boldsymbol{\Theta})^{-1} \frac{\partial \Omega_i(\boldsymbol{\Theta})}{\partial \theta} + \Omega_i(\boldsymbol{\Theta})^{-1} \frac{\partial \Omega_i(\boldsymbol{\Theta})}{\partial \theta'} \Omega_i(\boldsymbol{\Theta})^{-1} \frac{\partial^2 \Omega_i(\boldsymbol{\Theta})}{\partial \theta \partial \theta''} \right) \\
&\qquad \left. + \frac{\partial \mu(\boldsymbol{\Theta}, \boldsymbol{X}_i)}{\partial \theta} \Omega_i(\boldsymbol{\Theta})^{-1} \frac{\partial \Omega_i(\boldsymbol{\Theta})}{\partial \theta''} \Omega_i(\boldsymbol{\Theta})^{-1} \frac{\partial \mu(\boldsymbol{\Theta}, \boldsymbol{X}_i)}{\partial \theta'}^{\intercal} \right)
\end{aligned}$$

Appendix B Likelihood ratio test with the REML criterion

The blue term of Equation A in the log-likelihood is invariant to re-parameterisation while the red term is not. This means that a re-parametrisation of X into $\tilde{X} = BX$ with B invertible would not change the likelihood when using ML but would decrease the log-likelihood by $\log(|B|)$ when using REML.

```
LMMstar.options(optimizer = "FS",
  param.optimizer = c(n.iter = 1000, tol.score = 1e-3, tol.param = 1e-5))
```

Let's take an example:

```
## data(gastricbypassL, package = "LMMstar")
dfTest <- gastricbypassL
dfTest$glucagon2 <- dfTest$glucagon*2
```

where we multiply one column of the design matrix by 2. As mentionned previously this does not affect the log-likelihood when using ML:

```
logLik(lmm(weight ~ glucagon, data = dfTest, structure = UN(~time|id), method = "ML"))
logLik(lmm(weight ~ glucagon2, data = dfTest, structure = UN(~time|id), method = "ML"))
```

```
[1] -245.7909
[1] -245.7909
```

but it does when using REML:

```
logLik(lmm(weight ~ glucagon, data = dfTest, structure = UN(~time|id), method = "REML"))
logLik(lmm(weight ~ glucagon2, data = dfTest, structure = UN(~time|id), method = "REML"))
log(2)
```

```
[1] -245.0382
[1] -245.7313
[1] 0.6931472
```

Therefore, when comparing models with different mean effects there is a risk that the difference (or part of it) in log-likelihood is due to a new parametrisation and no only to a difference in model fit. This would typically be the case when adding an interaction where we can have a smaller restricted log-likehood when considering a more complex model:

```
set.seed(10)
dfTest$ff <- rbinom(NROW(dfTest), size = 1, prob = 0.5)
logLik(lmm(weight ~ glucagon, data = dfTest, structure = UN(~time|id), method = "REML"))
logLik(lmm(weight ~ glucagon*ff, data = dfTest, structure = UN(~time|id), method = "REML"))
```

```
[1] -245.0382
[1] -239.2056
```

This is quite counter-intuitive as more complex model should lead to better fit and would never happen when using ML:

```
logLik(lmm(weight ~ glucagon, data = dfTest, structure = UN(~time|id), method = "ML"))
logLik(lmm(weight ~ glucagon*ff, data = dfTest, structure = UN(~time|id), method = "ML"))
```

```
[1] -245.7909
[1] -237.3642
```

This is why, unless one knows what he/she is doing, it is not recommanded to use likelihood ratio test to assess relevance of mean parameters in mixed models estimated with REML.