Overview of the package LMMstar

Brice Ozenne

May 11, 2024

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: $\boldsymbol{Y} = (Y_1, \ldots, Y_T)$ where T can be for example be time (chronological ordering of the repetitions) or brain region (arbitrary ordering of the repetitions). Denoting by $\boldsymbol{X}$ the associated covariates and $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_T)$, the model can be written:

$$\boldsymbol{Y} = \boldsymbol{X}\beta + \boldsymbol{\varepsilon} \text{ where } \varepsilon \sim \mathcal{N}(0, \Omega)$$

where β are the mean parameters and the residual variance-covariance matrix, Ω, depends on a set of variance-covariance parameters (say γ) distinct of β. Key assumptions are:

- we observe n independent replicates of $\mathcal{O} = (\boldsymbol{Y}, \boldsymbol{X})$, i.e. at the cluster level, observations $(\mathcal{O}_1, \ldots, \mathcal{O}_n)$ are independent. The replicates should also be identically distributed up to a categorical variable (called strata variable in the following).
- the residual variance is independent of the mean value.

Additional assumptions are necessary in presence of missing values, typically correct specification of the conditional mean to have consistent estimates of the mean parameters. This case will sometimes be examplified by considering that only last outcome may be missing: the conditional mean $\mathbb{E}\left[Y_T | Y_1, Y_2, \ldots, Y_{T-1}\right]$ is then abreviated as $\mathbb{E}\left[Y_T | Y_{T-1}\right]$. Note that we do not require the residuals to be normally distributed to have valid estimates or statistical inference in large samples.

To get start, one should load the **LMMstar** package in the R session:

```
library(LMMstar)
```

This 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 for this overview is:

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

```
[1] '1.1.0'
```

It is recommanded to also the following packages, as some of the methods implemented in the package are relative to a generic method implmented in other packages:

```
library(ggplot2) ## autoplot method
library(nlme) ## ranef method
library(lava) ## iid, information, manifest methods
```

The user interface of the **LMMstar** package is made of the following functions:

- functions to describe or visualize the dataset:
  - `scatterplot` to visualize the marginal and bivariate distribution of continuous variables.
  - `summarize` to compute summary statistics, possibly stratified on a categorical variable.
  - `summarizeNA` to identify missing data patterns.
  - `partialCor` to compute partial correlation between two variables.
- the function `mt.test` to perform multiple Student's t-Tests and adjust the results for multiple testing.
- 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 perform Wald tests, i.e. test linear combinations of coefficients ($\widehat{\beta}_1 + \widehat{\beta}_2 = 0$ or $\widehat{\beta}_1 = \widehat{\beta}_2 = 0$). The output obtained with different `lmm` can be combined using `rbind`.
  - `coef` to extract the estimated model parameters ($\widehat{\beta}$ and possibly $\widehat{\gamma}$).
  - `confint` to extract the estimates with their confidence intervals.
  - `effects` to evaluate marginal effects, e.g. $\mathbb{E}\left[\mathbb{E}\left[Y|X_1 = 1\right] - \mathbb{E}\left[Y|X_1 = 0\right]\right]$ when $\boldsymbol{X} = (X_1, X_2)$.
  - `estimate` to test non-linear combinations of coefficients (Wald test via a first order delta method, e.g. $\widehat{\beta}_1/\widehat{\beta}_2 = 1$).
  - `fitted` to output the fitted mean ($X\widehat{\beta}$) or the conditional mean for observations with missing outcome (e.g. $X\widehat{\beta} + \widehat{\mathbb{E}}[\varepsilon_T|\varepsilon_{-T}]$).
  - `iid` to extract the influence function of the estimated parameters (φ), which satisfies $\sqrt{n}(\widehat{\beta} - \beta) = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \varphi\left(\mathcal{O}_i\right) + o_p(1)$
  - `levels` to extract the reference level for the mean structure. (i.e. what `(Intercept)` refers to in presence of categorical. covariates).
  - `logLik` to output the log-likelihood of the estimated model.
  - `model.tables` to extract the estimates, standard errors, p-value, and confidence intervals.
  - `plot` to obtain a diagnostic plots, partial residual plots, or a graphical display of the fitted values.
  - `predict` to compute the mean conditional on covariates and possible outcome values.
  - `profile` to display the likelihood or profile likelihood of the model.
  - `resample` to use non-parametric bootstrap or permutation test for statistical inference.
  - `residuals` to extract the observed residuals of the fitted model, possibly normalized ($\widehat{\Omega}^{-\frac{1}{2}}\widehat{\varepsilon}$).
  - `sigma` to extract the modeled residual variance covariance matrix ($\widehat{\Omega}$).
  - `summary` to obtain a summary of the input, model fit, and estimated values.
  - `vcov` to extract the variance-covariance matrix of the mean parameters ($\widehat{\Sigma}_{\widehat{\beta}}$).
- the `mlmm` function to fit group-specific linear mixed models and gather the estimated coefficients.
- 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.

1 Illustrative dataset

To illustrate the functionalities of the package, we will use the gastricbypass dataset. The long format can be imported using:

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

```
  id visit time weight glucagonAUC
1  1     1  -13  127.2      20.690
2  2     1  -13  165.2      49.922
3  3     1  -13  109.7      42.434
4  4     1  -13  146.2      27.517
5  5     1  -13  113.1      29.151
6  6     1  -13  158.8      42.700
```

See `?gastricbypassL` for a presentation of the dataset. It is convenient to encode the time variable in two formats:

- numeric, e.g. here with the time in week since surgery (`time` variable taking values -13,-1,1,13 - negative times refering to before an intervention and positive times after the intervention).
- factor, e.g. here with the visit index (`visit` variable taking value 1,2,3,4)

To illustrate certain functionalities we will use an (artificial) group variable:

```
gastricbypassL$group <- as.factor(as.numeric(gastricbypassL$id)%%2)
```

and dichotomize time as before and after the intervention:

```
gastricbypassL$baseline <- gastricbypassL$time<0
```

The corresponding wide format is

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

```
  id weight1 weight2 weight3 weight4 glucagonAUC1 glucagonAUC2 glucagonAUC3 glucagonAUC4
1  1   127.2   120.7   115.5   108.1       20.690       20.535       92.600       43.434
2  2   165.2   153.4   149.2   132.0       49.922       58.513       49.633       35.747
3  3   109.7   101.6    97.7    87.1       42.434       25.770       91.240       83.137
4  4   146.2   142.4   136.7   123.0       27.517       27.552       59.360       21.371
5  5   113.1   105.6    99.9    87.7       29.151           NA       86.859       57.970
6  6   158.8   143.6   134.6   108.7       42.700       31.616       53.408       37.636
```

for which we can also add the group variable:

```
gastricbypassW$group <- as.numeric(gastricbypassW$id)%%2
```

In some cases we will, for comparison, perform complete case analyses with the following dataset:

```
gastricbypassL.NNA <- gastricbypassL[!is.na(gastricbypassL$glucagonAUC),]
```

2 Visualization & descriptive statistics

2.1 Graphical display

A scatterplot of the data can obtained by specifying which columns to display when using the wide format:

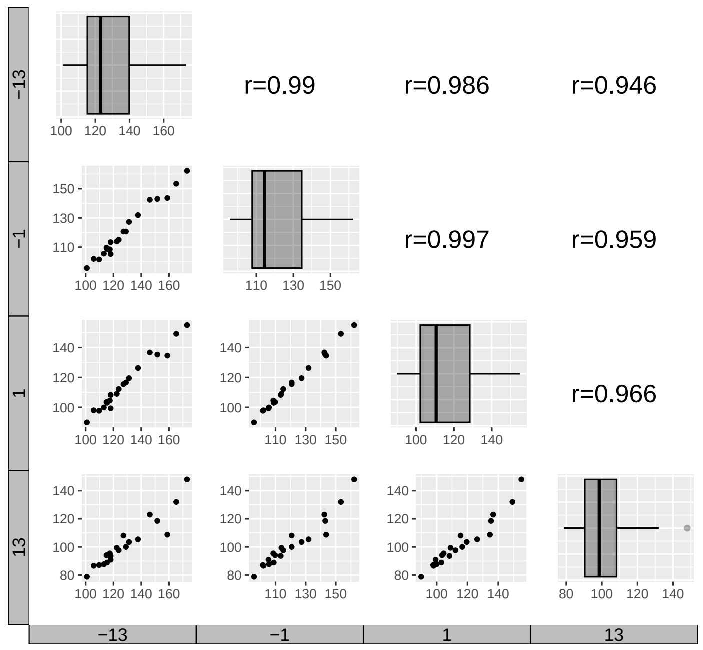
```
scatterplot(gastricbypassW, ## left panel
            columns = c("weight1","weight2","weight3","weight4"))
```

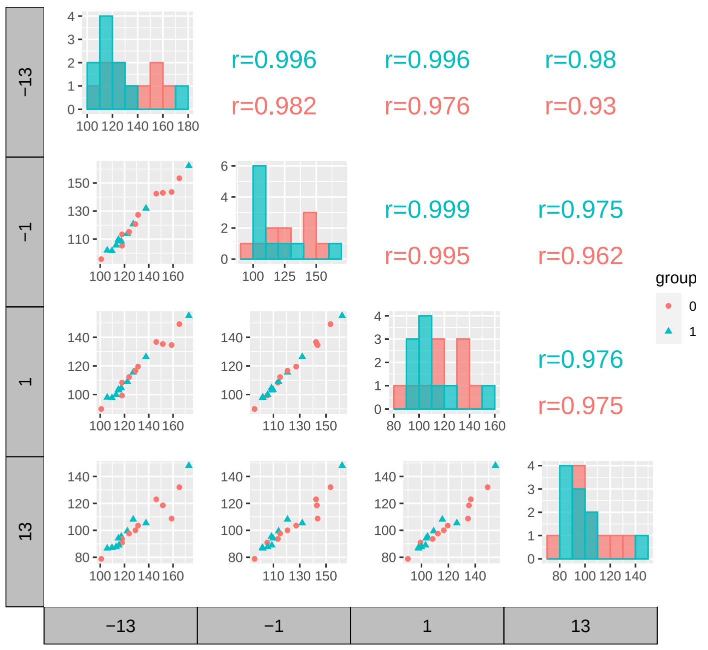
When using the long format, a formula should describe the structure of the data: `outcome ~ order|cluster`

- the left hand side indicates the values to be displayed (here weight)
- the right hand side indicates the ordering of the repetitions (here over time) and how the repetitions are grouped within clusters (here within subject).

When calling `scatterplot`, the argument `group` leads to different color per group and the argument `type.diag` enables to use histograms (or density plots) instead of boxplots:

```
scatterplot(weight~time|id, data = gastricbypassL, ## right panel
            type.diag = "hist", group = "group")
```





By default the resulting object will be of class `list`. A `ggplot2` object can be obtained by setting the argument `facet` to `"grid2"`. This requires to have installed the package ggh4x and will produce a slightly different graphical display.

There is (currently) not dedicated function to obtain spaghetti plots. Instead one can use the ggplot2 package with the long format, e.g.:

```
gg.spa <- ggplot(gastricbypassL, aes(x=time,y=weight,group=id,color=id))
gg.spa <- gg.spa + geom_point() + geom_line()
gg.spa
```

2.2 Missing data patterns

The `summarizeNA` function identifies the possible combinations of observed/missing data:

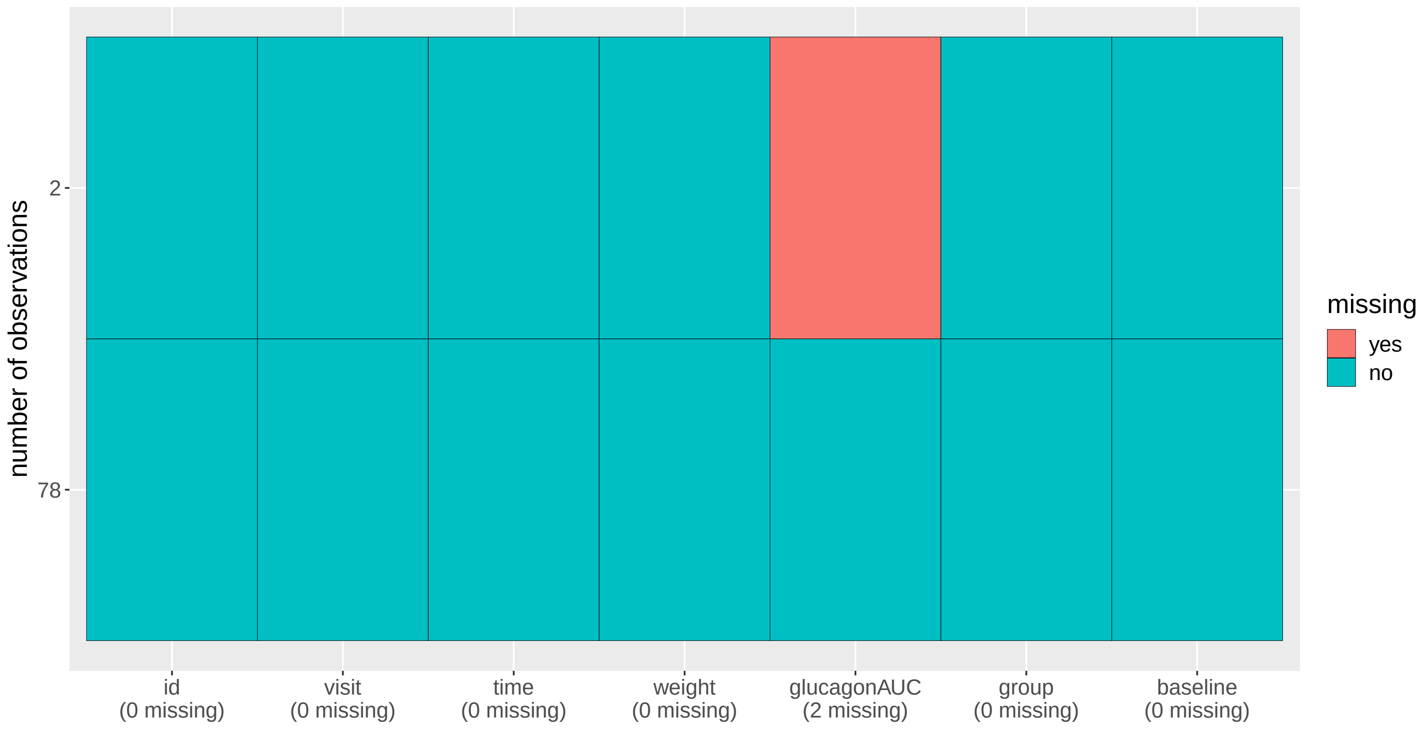
```
mp <- summarizeNA(gastricbypassL)
mp
```

```
frequency missing.pattern n.missing id visit time weight glucagonAUC group baseline
       78         0000000         0  0     0    0      0           0     0        0
        2         0000100         1  0     0    0      0           1     0        0
```

A graphical representation can be obtained using `plot`:

```
plot(mp)
```

See `help(plot.summarizeNA)` for options to customize the graphical display.



2.3 Summary statistics

Mean, standard deviation, and other summary statistic can be computed with respect to a categorical variable (typically time) using the `summarize` function:
(⚠ this function has the same name as a function from the dplyr package. If you have loaded dplyr, you should use `LMMstar::summarize`)

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

```
      outcome time observed missing  mean   sd    min    q1 median    q3   max
1      weight  -13       20       0 129.0 20.3 100.90 115.3  123.1 139.8 173.0
2               -1       20       0 121.2 18.9  95.70 107.8  114.5 134.5 162.2
3                1       20       0 115.7 18.3  89.90 102.2  110.6 128.4 155.0
4               13       20       0 102.4 17.1  78.80  90.4   98.5 108.2 148.0
5 glucagonAUC  -13       20       0  32.3 15.5  10.28  21.3   27.9  42.5  69.1
6               -1       19       1  29.7 13.7   9.87  21.2   25.8  33.6  67.7
7                1       19       1  76.9 27.9  35.85  56.5   73.8  91.9 135.9
8               13       20       0  52.0 21.0  21.37  37.2   51.2  57.9 109.2
```

Specifying a cluster (`id`) and ordering variable (`time`) enable to output correlation matrices:
(⚠ there should be no duplicated value of the ordering variable within cluster)

```
sss2 <- summarize(weight ~ time|id, data = gastricbypassL, na.rm = TRUE)
print(sss2, digits = 3)
```

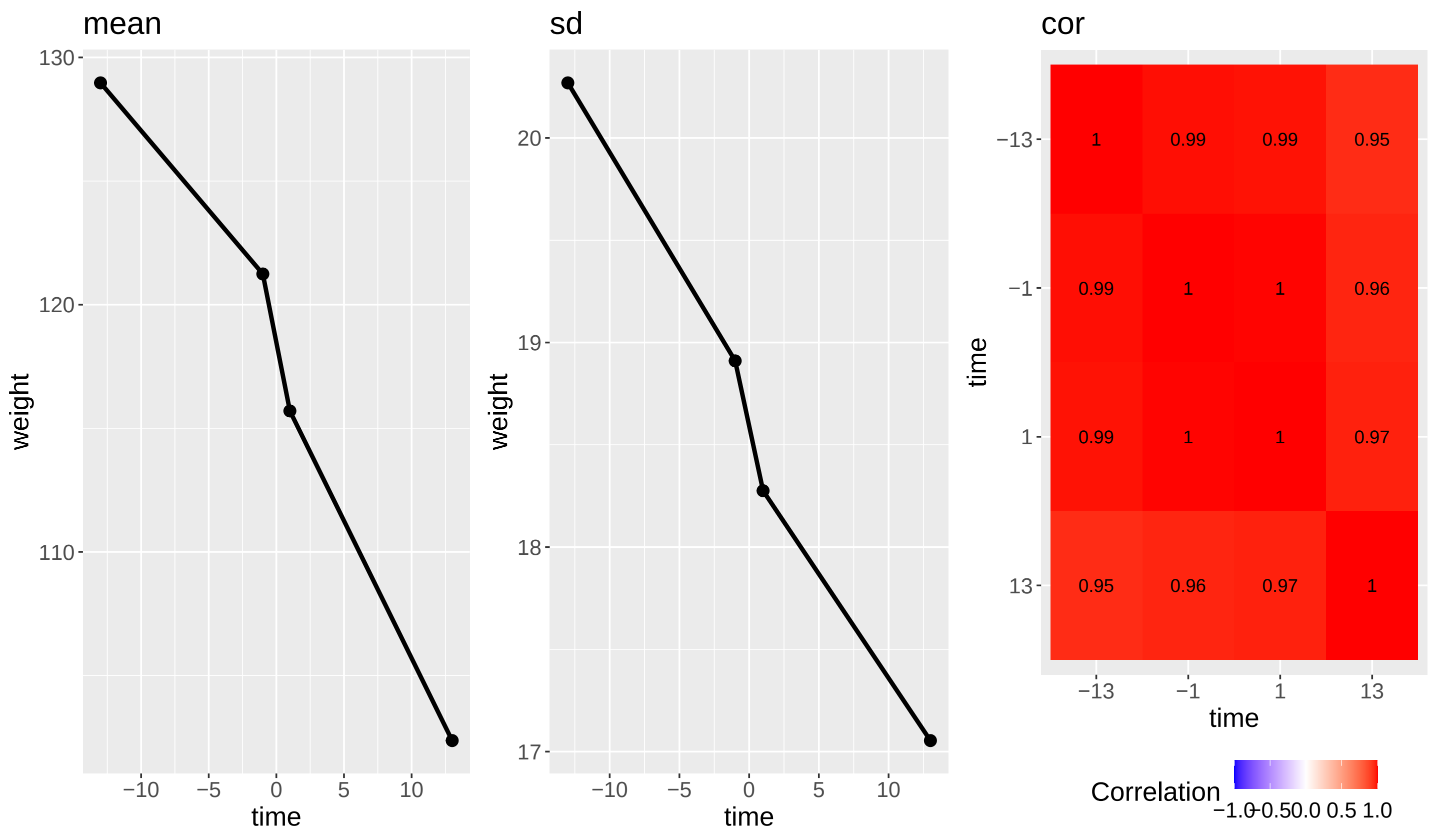
```
  time observed missing mean   sd   min    q1 median  q3 max
1  -13       20       0  129 20.3 100.9 115.3  123.1 140 173
2   -1       20       0  121 18.9  95.7 107.8  114.5 135 162
3    1       20       0  116 18.3  89.9 102.2  110.6 128 155
4   13       20       0  102 17.1  78.8  90.4   98.5 108 148

 Pearson's correlation:
      -13    -1     1    13
-13 1.000 0.990 0.986 0.946
-1  0.990 1.000 0.997 0.959
1   0.986 0.997 1.000 0.966
13  0.946 0.959 0.966 1.000
```

Graphical displays of the summary statistics can be obtained via the `plot` method, where the argument `type` specifies the summary statistic to be displayed:

```
plot(sss2, type = "mean") ## left panel
plot(sss2, type = "sd") ## middle panel
plot(sss2, type = "cor") ## right panel
```

See `help(plot.summarize)` for options to customize the graphical display.


mean
weight
130
120
110
time
−10
−5
0
5
10
sd
weight
20
19
18
17
time
−10
−5
0
5
10
cor
time
−13
−1
1
13
1 0.99 0.99 0.95
0.99 1 1 0.96
0.99 1 1 0.97
0.95 0.96 0.97 1
time
−13
−1
1
13
Correlation
−1.0 −0.5 0.0 0.5 1.0

2.4 Correlation and partial correlations

The partialCor function can be used to evaluate group-specific correlations, e.g.:

```
partialCor(weight + glucagonAUC ~ 1, by = "group", data = gastricbypassL)
```

```
                                estimate    se   df  lower    upper p.value
0: rho(weight,glucagonAUC)   -0.328 0.143 21.8 -0.587 -0.00886  0.0447
1: rho(weight,glucagonAUC)   -0.354 0.141 22.5 -0.607 -0.03631  0.0313
```

This willl lead to the same estimate as the cor.test function (Pearson correlation):

```
gastricbypassL.0 <- gastricbypassL[gastricbypassL$group==0,]
rho <- cor.test(gastricbypassL.0$weight, gastricbypassL.0$glucagonAUC)
c(rho$estimate, p.value = rho$p.value)
```

```
      cor   p.value
-0.328481  0.038505
```

However the p-value may differ, especially in small samples, as partialCor uses a different (and probably more crude) small sample approximation for the estimator's distribution. Nevertheless partialCor enables to compare correlation coefficients across groups, by specifying the argument effects:

```
partialCor(weight + glucagonAUC ~ 1, by = "group", effects = "Dunnett",
           data = gastricbypassL)
```

```
                                                     estimate se df lower upper p.value
1:rho(weight,glucagonAUC) - 0:rho(weight,glucagonAUC)  -0.0255 NA NA    NA    NA   0.899
```

Partial correlations can be also computed by specifying covariate to adjust for on the right-hand side:

```
partialCor(weight4 + glucagonAUC4 ~ weight1,
           data = gastricbypassW)
```

```
                          estimate    se   df  lower upper p.value
rho(weight4,glucagonAUC4)    0.112 0.233 9.12 -0.397 0.568   0.645
```

When the set of covariates is outcome-dependent, a list of formulas can be used instead:

```
partialCor(list(weight1 ~ glucagonAUC1, weight4 ~ glucagonAUC4),
           data = gastricbypassW)
```

```
                     estimate     se   df lower upper  p.value
rho(weight1,weight4)    0.946 0.0252 26.4 0.861 0.979 5.51e-08
```

These partial correlations are defined as the residual correlation between the outcomes, i.e. the correlation once the covariate effects have been substracted from the outcome, and a linear mixed model is used to estimated them.

3 Multiple Student's t-tests

When working with multiple outcomes and having no missing data, mean comparisons between exposure groups can be carried out using Student's t-tests at each timepoint, e.g.:

```
restt <- t.test(weight1 ~ group, data = gastricbypassW)
c(estimate = unname(diff(restt$estimate)), p.value = restt$p.value)
```

```
 estimate   p.value
-10.60000   0.25282
```

And so on for the three other timepoints. Morever results would typically need to be adjusted for multiple comparisons, e.g. when looking for any mean difference. This can be faciliated by

```
## single step max-test adjustment (see help(confint.Wald_lmm) for details)
mt.test(weight1+weight2+weight3+weight4~group, data = gastricbypassW)
```

```
       by parameter estimate     se     df   lower   upper p.value
1 weight1     group   -10.60 8.9717 17.965 -30.968  9.7680 0.31894
2 weight2     group    -9.50 8.3951 17.985 -28.559  9.5590 0.34164
3 weight3     group    -8.92 8.1295 17.959 -27.376  9.5358 0.35891
4 weight4     group    -4.59 7.7607 17.682 -22.209 13.0286 0.66331
```

The method used to adjust confidence intervals and p-values for multiple comparisons can be specified via the `method` argument, e.g.:

```
## no adjustment
mt.test(weight1+weight2+weight3+weight4~group, data = gastricbypassW, method = "none")
```

```
       by parameter estimate     se     df   lower   upper p.value
1 weight1     group   -10.60 8.9717 17.965 -29.452  8.2516 0.25281
2 weight2     group    -9.50 8.3951 17.985 -27.139  8.1386 0.27266
3 weight3     group    -8.92 8.1295 17.959 -26.002  8.1622 0.28703
4 weight4     group    -4.59 7.7607 17.682 -20.916 11.7356 0.56171
```

```
## bonferroni adjustment
mt.test(weight1+weight2+weight3+weight4~group, data = gastricbypassW, method = "bonferroni")
```

```
       by parameter estimate     se     df   lower  upper p.value
1 weight1     group   -10.60 8.9717 17.965 -35.498 14.298       1
2 weight2     group    -9.50 8.3951 17.985 -32.795 13.795       1
3 weight3     group    -8.92 8.1295 17.959 -31.481 13.641       1
4 weight4     group    -4.59 7.7607 17.682 -26.165 16.985       1
```

4 Linear mixed model (LMM)

4.1 Classical covariance patterns

Several build-in covariance patterns can be used when specifying the linear model. The most basic ones are the **identity** structure:

```r
eId.lmm <- lmm(glucagonAUC ~ visit*group, repetition = ~time|id,
               structure = "ID", data = gastricbypassL)
eId.lmm
cat(" modeled residual variance-covariance: \n");sigma(eId.lmm)
```

```
                Linear regression

 outcome/cluster/time: glucagonAUC/id/time
 data                : 78 observations from 20 clusters
 parameter           : 8 mean ((Intercept) visit2 visit3 visit4 group1 visit2:group1 visit3:group1
                       1 variance (sigma)
 log-restr.likelihood: -316.461119970244
 convergence         : TRUE (0 iterations)
 modeled residual variance-covariance:
       -13     -1      1     13
-13 381.35   0.00   0.00   0.00
-1    0.00 381.35   0.00   0.00
1     0.00   0.00 381.35   0.00
13    0.00   0.00   0.00 381.35
```

and the **independence** structure:

```r
eInd.lmm <- lmm(glucagonAUC ~ visit*group, repetition = ~time|id,
                structure = "IND", data = gastricbypassL)
eInd.lmm
cat(" modeled residual variance-covariance: \n");sigma(eInd.lmm)
```

```
                Linear regression with heterogeneous residual variance

 outcome/cluster/time: glucagonAUC/id/time
 data                : 78 observations from 20 clusters
 parameter           : 8 mean ((Intercept) visit2 visit3 visit4 group1 visit2:group1 visit3:group1
                       4 variance (sigma k.-1 k.1 k.13)
 log-restr.likelihood: -310.428096419287
 convergence         : TRUE (0 iterations)
 modeled residual variance-covariance:
       -13     -1      1     13
-13 209.44   0.00   0.00   0.00
-1    0.00 174.81   0.00   0.00
1     0.00   0.00 768.23   0.00
13    0.00   0.00   0.00 382.95
```

The most common linear mixed model uses a **compound symmetry** structure:

```
eCS.lmm <- lmm(glucagonAUC ~ visit*group, repetition = ~time|id,
               structure = "CS", data = gastricbypassL)
eCS.lmm
cat(" modeled residual variance-covariance: \n");sigma(eCS.lmm)
```

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

 outcome/cluster/time: glucagonAUC/id/time
 data                : 78 observations from 20 clusters
 parameter           : 8 mean ((Intercept) visit2 visit3 visit4 group1 visit2:group1 visit3:group1
                       1 variance (sigma)
                       1 correlation (rho(id))
 log-restr.likelihood: -314.394203759159
 convergence         : TRUE (6 iterations)
 modeled residual variance-covariance:
        -13      -1       1      13
-13 380.580  82.741  82.741  82.741
-1   82.741 380.580  82.741  82.741
1    82.741  82.741 380.580  82.741
13   82.741  82.741  82.741 380.580
```

A more flexible model can be obtained with a **toeplitz** covariance matrix:

```
eTOE.lmm <- lmm(glucagonAUC ~ visit*group, repetition = ~time|id,
                structure = "TOEPLITZ", data = gastricbypassL)
eTOE.lmm
cat(" modeled residual correlation: \n");cov2cor(sigma(eTOE.lmm))
```

```
                Linear Mixed Model with a block Toeplitz covariance matrix

 outcome/cluster/time: glucagonAUC/id/time
 data                : 78 observations from 20 clusters
 parameter           : 8 mean ((Intercept) visit2 visit3 visit4 group1 visit2:group1 visit3:group1
                       4 variance (sigma k.-1 k.1 k.13)
                       4 correlation (rho(12) rho(14) rho(26) rho(2))
 log-restr.likelihood: -297.525485582536
 convergence         : TRUE (15 iterations)
 modeled residual correlation:
          -13       -1        1        13
-13  1.000000 0.700020 0.093615 -0.082963
-1   0.700020 1.000000 0.016795  0.093615
1    0.093615 0.016795 1.000000  0.700020
13  -0.082963 0.093615 0.700020  1.000000
```

And an even more flexible model can be obtained with an **unstructured** covariance matrix:

```
eUN.lmm <- lmm(glucagonAUC ~ visit*group, repetition = ~time|id,
               structure = "UN", data = gastricbypassL)
eUN.lmm
cat(" modeled residual variance-covariance: \n");sigma(eUN.lmm)
```

```
                 Linear Mixed Model with an unstructured covariance matrix

 outcome/cluster/time: glucagonAUC/id/time
 data                : 78 observations from 20 clusters
 parameter           : 8 mean ((Intercept) visit2 visit3 visit4 group1 visit2:group1 visit3:group1
                       4 variance (sigma k.-1 k.1 k.13)
                       6 correlation (rho(-13,-1) rho(-13,1) rho(-13,13) rho(-1,1) rho(-1,13) rho(1
 log-restr.likelihood: -295.314056198772
 convergence         : TRUE (8 iterations)
 modeled residual variance-covariance:
        -13      -1        1      13
-13 209.442 150.2502 106.4000 -24.202
-1  150.250 168.1138   1.3064 -23.884
1   106.400   1.3064 748.0769 288.184
13  -24.202 -23.8844 288.1839 382.952
```

Stratification of the covariance structure on a categorical variable is also possible:

- e.g. to get a **stratified compound symmetry**

```
eSCS.lmm <- lmm(glucagonAUC ~ visit*group, repetition = ~time|id,
                structure = CS(group~1), data = gastricbypassL)
eSCS.lmm
```

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

outcome/cluster/time: glucagonAUC/id/time
data                : 78 observations from 20 clusters
parameter           : 8 mean ((Intercept) visit2 visit3 visit4 group1 visit2:group1 visit3:group1 v
                      2 variance (sigma:0 sigma:1)
                      2 correlation (rho(id):0 rho(id):1)
log-restr.likelihood: -314.123797063042
convergence         : TRUE (7 iterations)
```

- e.g. **stratified unstructured** covariance matrix:

```
eSUN.lmm <- lmm(glucagonAUC ~ visit*group, repetition = ~time|id,
                structure = UN(group~1), data = gastricbypassL)
eSUN.lmm
```

```
                 Linear Mixed Model with a stratified unstructured covariance matrix

outcome/cluster/time: glucagonAUC/id/time
data                : 78 observations from 20 clusters
parameter           : 8 mean ((Intercept) visit2 visit3 visit4 group1 visit2:group1 visit3:group1 v
                      8 variance (sigma:0 sigma:1 k.-1:0 k.1:0 k.13:0 k.-1:1 k.1:1 k.13:1)
                      12 correlation (rho(-13,-1):0 rho(-13,1):0 rho(-13,13):0 rho(-1,1):0 rho(-1,1
log-restr.likelihood: -286.536815485471
convergence         : TRUE (10 iterations)
```

with modeled residual variance-covariance:

sigma(eSCS.lmm)	sigma(eSUN.lmm)

```
$`0`
        -13      -1       1      13
-13 334.289  50.782  50.782  50.782
-1   50.782 334.289  50.782  50.782
1    50.782  50.782 334.289  50.782
13   50.782  50.782  50.782 334.289

$`1`
       -13     -1      1     13
-13 428.46 115.09 115.09 115.09
-1  115.09 428.46 115.09 115.09
1   115.09 115.09 428.46 115.09
13  115.09 115.09 115.09 428.46
```

```
$`0`
        -13      -1       1      13
-13 309.85 251.512 102.189 -42.250
-1  251.51 274.752 -79.811 -90.718
1   102.19 -79.811 579.110 163.767
13  -42.25 -90.718 163.767 173.439

$`1`
          -13     -1       1       13
-13 109.0309 48.667 104.908  -6.1549
-1   48.6665 59.395  93.976  43.2144
1   104.9077 93.976 967.583 450.8899
13   -6.1549 43.214 450.890 592.4655
```

Finally the some covariance patterns like the compound symmetry structure may depend on covariates:

- e.g. to obtain a **block compound symmetry** structure[1]:

```
eBCS.lmm <- lmm(glucagonAUC ~ visit*group, repetition = ~time|id,
                structure = CS(~baseline, type = "homogeneous"), data = gastricbypassL)
eBCS.lmm
cat(" modeled residual variance-covariance: \n");sigma(eBCS.lmm)
```

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

 outcome/cluster/time: glucagonAUC/id/time
 data                : 78 observations from 20 clusters
 parameter           : 8 mean ((Intercept) visit2 visit3 visit4 group1 visit2:group1 visit3:group1
                       1 variance (sigma)
                       2 correlation (rho(id/baseline) rho(id))
 log-restr.likelihood: -308.994835006264
 convergence         : TRUE (6 iterations)
 modeled residual variance-covariance:
        -13      -1       1      13
-13 380.957 226.403  15.465  15.465
-1  226.403 380.957  15.465  15.465
1    15.465  15.465 380.957 226.403
13   15.465  15.465 226.403 380.957
```

- e.g. to obtain a **block unstructured** covariance matrix:

```
eBUN.lmm <- lmm(glucagonAUC ~ visit*group, repetition = ~time|id,
                structure = CS(~baseline, type = "heterogeneous"), data = gastricbypassL)
eBUN.lmm
cat(" modeled residual variance-covariance: \n");sigma(eBUN.lmm)
```

```
                Linear Mixed Model with a block unstructured covariance matrix

 outcome/cluster/time: glucagonAUC/id/time
 data                : 78 observations from 20 clusters
 parameter           : 8 mean ((Intercept) visit2 visit3 visit4 group1 visit2:group1 visit3:group1
                       2 variance (sigma k.TRUE)
                       3 correlation (rho(FALSE) rho(FALSE,TRUE) rho(TRUE))
 log-restr.likelihood: -300.047474124556
 convergence         : TRUE (7 iterations)
 modeled residual variance-covariance:
        -13      -1       1      13
-13 189.420 150.356  15.353  15.353
-1  150.356 189.420  15.353  15.353
1    15.353  15.353 570.908 300.071
13   15.353  15.353 300.071 570.908
```

[1]similar to nested random effects

4.2 User-specific covariance patterns

It is possible input user-specific covariance patterns under the following model for the residuals:

$$\Omega = \boldsymbol{\sigma}^\intercal R \boldsymbol{\sigma}$$

- $\boldsymbol{\sigma} = f(\boldsymbol{\theta}_\sigma, Z_\sigma)$ is a vector of residual standard errors depending on a vector of parameters $\boldsymbol{\theta}_\sigma$ and possible covariates via the design matrix Z_σ.
- $R = g(\boldsymbol{\theta}_R, Z_R)$ is a matrix of residual correlations depending on a vector of parameters $\boldsymbol{\theta}_R$ and possible covariates via the design matrix Z_R.

To be more concrete, consider the following correlation matrix

```
rho.2block <- function(p,n.time,X){
  rho <- matrix(1, nrow = n.time, ncol = n.time)
  rho[1,2] <- rho[2,1] <- rho[4,5] <- rho[5,4] <- p["rho1"]
  rho[1,3] <- rho[3,1] <- rho[4,6] <- rho[6,4] <- p["rho2"]
  rho[2,3] <- rho[3,2] <- rho[5,6] <- rho[6,5] <- p["rho3"]
  rho[4:6,1:3] <- rho[1:3,4:6] <- p["rho4"]
  return(rho)
}
Rho <- rho.2block(p = c(rho1=0.25,rho2=0.5,rho3=0.4,rho4=0.1),
                  n.time = 6)
Rho
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,] 1.00 0.25  0.5 0.10 0.10  0.1
[2,] 0.25 1.00  0.4 0.10 0.10  0.1
[3,] 0.50 0.40  1.0 0.10 0.10  0.1
[4,] 0.10 0.10  0.1 1.00 0.25  0.5
[5,] 0.10 0.10  0.1 0.25 1.00  0.4
[6,] 0.10 0.10  0.1 0.50 0.40  1.0
```

and the corresponding dataset:

```
set.seed(11)
Y <- mvtnorm::rmvnorm(1000, mean = rep(0,6), sigma = Rho)
dfW <- cbind(id = 1:NROW(Y), as.data.frame(Y))
dfL <- reshape2::melt(dfW, id.vars = "id", variable.name = "time")
dfL[dfL$id %in% 1:2,]
```

```
     id time      value                  id time      value
1     1   V1 -0.9842079            2      2   V1  1.2402726
1001  1   V2 -0.3681245            1002   2   V2  0.6494215
2001  1   V3 -1.6174652            2002   2   V3  0.3272105
3001  1   V4 -1.4994103            3002   2   V4 -1.0626973
4001  1   V5  0.7493107            4002   2   V5 -0.9013244
5001  1   V6 -1.0719657            5002   2   V6 -0.6696714
```

To estimate the corresponding mixed model we first define a new covariance structure:

```
myStruct <- CUSTOM(~time,
                   FCT.sigma = function(p,n.time,X){rep(p,n.time)}, ## function f
                   init.sigma = c("sigma"=1),
                   FCT.rho = rho.2block, ## function g
                   init.rho = c("rho1"=0.25,"rho2"=0.25,"rho3"=0.25,"rho4"=0.25))
```

and then call `lmm` with this structure structure:

```
e.lmmCUSTOM <- lmm(value~time, repetition=~time|id,
                   structure = myStruct, data=dfL,
                   df = FALSE) ## df = FALSE to save computation time
logLik(e.lmmCUSTOM)
```

```
[1] -7962.243
```

The optimization procedure may be slow but should eventually reaches an optimum. We can then output the estimated correlation matrix:

```
cov2cor(sigma(e.lmmCUSTOM))
```

```
           V1         V2         V3         V4         V5         V6
V1 1.00000000 0.24898095 0.50058994 0.09053785 0.09053785 0.09053785
V2 0.24898095 1.00000000 0.36110943 0.09053785 0.09053785 0.09053785
V3 0.50058994 0.36110943 1.00000000 0.09053785 0.09053785 0.09053785
V4 0.09053785 0.09053785 0.09053785 1.00000000 0.24898095 0.50058994
V5 0.09053785 0.09053785 0.09053785 0.24898095 1.00000000 0.36110943
V6 0.09053785 0.09053785 0.09053785 0.50058994 0.36110943 1.00000000
```

Comparison to build-in structure: consider the following model using a build-in compound symmetry structure:

```
system.time(
  e.lmmDEFAULT.CS <- lmm(value~time, repetition = ~time|id,
                         structure = "CS", data = dfL,
                         df = FALSE)
)
```

```
 user  system elapsed
0.097   0.000   0.097
```

Using instead `CUSTOM` to specifying this structure:

```
myCS <- CUSTOM(~1,
               FCT.sigma = function(p,n.time,X){rep(p,n.time)},
               init.sigma = c("sigma"=1),
               FCT.rho = function(p,n.time,X){p+diag(1-p,n.time,n.time)},
               init.rho = c("rho"=0.5))
```

is considerably slower than using the pre-specified structure:

```
system.time(
  e.lmmCUSTOM.CS <- lmm(value~time, repetition = ~time|id,
                        structure = myCS, data = dfL,
                        df = FALSE)
)
```

```
 user  system elapsed 
0.952   0.019   0.972 
```

but will lead to the same estimates:

```
logLik(e.lmmDEFAULT.CS)
logLik(e.lmmCUSTOM.CS)
```

```
[1] -8186.859
[1] -8186.859
```

There are two reasons for the slower execution time: slower evaluation of the derivatives (since they are obtained by numerical differentiation) and worse starting point, as reflected by the larger number of interations needed to reach convergence:

```
e.lmmDEFAULT.CS$opt$n.iter
e.lmmCUSTOM.CS$opt$n.iter
```

```
[1] 1
[1] 4
```

Faster execution time can be obtained by specifying the first and second derivative regarding each parameter:

```
myCS.wD <- CUSTOM(~1,
                  FCT.sigma = function(p,n.time,X){rep(p,n.time)},
                  dFCT.sigma = function(p,n.time,X){list(sigma = rep(1,n.time))},
                  d2FCT.sigma = function(p,n.time,X){list(sigma = rep(0,n.time))},
                  init.sigma = c("sigma"=1),
                  FCT.rho = function(p,n.time,X){p+diag(1-p,n.time,n.time)},
                  dFCT.rho = function(p,n.time,X){list(rho = 1-diag(1,n.time,n.time))},
                  d2FCT.rho = function(p,n.time,X){list(rho = matrix(0,n.time,n.time))},
                  init.rho = c("rho"=0.5))
```

```
system.time(
  e.lmmCUSTOMwD.CS <- lmm(value~time,
                          repetition = ~time|id,
                          structure = myCS.wD,
                          data = dfL, df = FALSE
                          )
)
```

```
 user  system elapsed 
0.699   0.004   0.703 
```

4.3 Estimation procedure

Initialiation: by default the mean parameters are initialized using Ordinary Least Squares (OLS) and the variance and correlation parameters are initialized by minimizing the difference between the observed and residuals variance-covariance matrix. These values can be visualized by specifying the argument `control`:

```
eCS.lmm.bis <- update(eCS.lmm, control = list(trace = 2))
```

```
Initialization:
  (Intercept)        visit2        visit3        visit4        group1 visit2:group1 visit3:group1
     38.72897      -4.73433      31.43303       4.52138     -12.82462       3.75946      27.00150
visit4:group1         sigma       rho(id)
     30.22391      19.52828       0.22819

Loop:
******
  (Intercept)        visit2        visit3        visit4        group1 visit2:group1 visit3:group1
     38.72897      -4.73433      31.43303       4.52138     -12.82462       3.80337      27.48103
visit4:group1         sigma       rho(id)
     30.22391      19.50846       0.21741
Convergence after 6 iterations: max score=1.2413e-05 | max change in coefficient=4.5167e-06
```

It is possible to input user-defined value:

- for all parameters (vector)

```
init.all <- coef(eCS.lmm, effects = "all")
eCS.lmm.bis <- update(eCS.lmm, control = list(init = init.all, trace = 1))
```

```
Convergence after 0 iteration: max score=1.2413e-05
```

- the mean parameters only (vector)

```
init.mean <- coef(eCS.lmm, effects = "mean")
eCS.lmm.bis <- update(eCS.lmm, control = list(init = init.mean, trace = 2))
```

```
Initialization:
  (Intercept)        visit2        visit3        visit4        group1 visit2:group1 visit3:group1
     38.72897      -4.73433      31.43303       4.52138     -12.82462       3.80337      27.48103
visit4:group1         sigma       rho(id)
     30.22391      19.52904       0.22849

Loop:
******
  (Intercept)        visit2        visit3        visit4        group1 visit2:group1 visit3:group1
     38.72897      -4.73433      31.43303       4.52138     -12.82462       3.80337      27.48103
visit4:group1         sigma       rho(id)
     30.22391      19.50846       0.21741
Convergence after 6 iterations: max score=1.4893e-05 | max change in coefficient=5.3866e-06
```

- a full data variance-covariance matrix (matrix).

```
init.vcov <- sigma(eCS.lmm)
eCS.lmm.bis <- update(eCS.lmm, control = list(init = init.vcov, trace = 1))
```

```
Convergence after 0 iteration: max score=1.2413e-05
```

Optimizer: by default the optimizer is a Newton Raphson algorithm with backtracking. At each iteration:

- it computes the first two moments (score, information) according to the current parameters values.
- it updates the variance-covariance parameters according to the gradient multiplied by the inverse of the information.
- it updates the mean parameters by generalized least squares (using the updated variance-covariance parameters).
- it checks whether the log-likelihoood at the updated estimates is well defined and higher than at the previous estimates. If this is not the case, the step is re-run with half the update of the variance-covariance parameters (backtracking).

One can modify the maximum number of iterations (`n.iter`), maximum number of backtracking steps (`n.backtracking`), the maximum score (absolute) value over all parameters (`tol.score`) and (absolute) maximum difference in parameter value between to iterations (`tol.param`) used to declare convergence. It is also possible to use another optimizer (`optimizer`). All these elements should be passed to the argument `control` of `lmm` using a list.

4.4 Model output

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

```
summary(eUN.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:

    $ glucagonAUC: num  20.7 49.9 42.4 27.5 29.2 ...
    $ visit      : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
    $ group      : Factor w/ 2 levels "0","1": 2 1 2 1 2 1 2 1 2 1 ...
    reference level: visit=1;group=0

Estimation procedure

  - Restricted Maximum Likelihood (REML)
  - log-likelihood :-295.31
  - parameters: mean = 8, variance = 4, correlation = 6
  - convergence: TRUE (8 iterations)
    largest |score| = 4.6771e-05 for rho(-1,1)
            |change|= 1.68033723859651e-05 for visit3:group1

Residual variance-covariance: unstructured

  - correlation structure: ~0 + time
            -13        -1        1       13
    -13   1.0000   0.80072 0.26880 -0.0855
    -1    0.8007   1.00000 0.00368 -0.0941
    1     0.2688   0.00368 1.00000  0.5384
    13   -0.0855 -0.09413 0.53842  1.0000

  - variance structure: ~time
              standard.deviation ratio
    sigma.-13               14.5 1.000
    sigma.-1                13.0 0.896
    sigma.1                 27.4 1.890
    sigma.13                19.6 1.352
```

```
Fixed effects: glucagonAUC ~ visit * group

                 estimate     se   df   lower  upper p.value
    (Intercept)    38.729  4.576   18  29.114 48.344 < 1e-04 ***
    visit2         -4.734  2.776 17.5 -10.577  1.109 0.10574
    visit3         31.433   8.63 17.6  13.272 49.594 0.00192  **
    visit4          4.521  8.005   18 -12.297  21.34 0.57917
    group1        -12.825  6.472   18 -26.422  0.773 0.06302   .
    visit2:group1   3.987  3.996 17.9   -4.41 12.383 0.33169
    visit3:group1  27.571  12.42 17.8   1.461 53.682 0.03963   *
    visit4:group1  30.224 11.321   18   6.439 54.008 0.01562   *
    --------------------------------------------------------
   Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1.
   Columns lower and upper contain 95% pointwise confidence intervals for each coefficient.
   Model-based standard errors are derived from the observed information (column se).
   Degrees of freedom were computed using a Satterthwaite approximation (column df).
```

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.

4.5 Extract estimated coefficients

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

```
coef(eUN.lmm)
```

```
  (Intercept)        visit2        visit3        visit4        group1 visit2:group1 visit3:group1
      38.7290       -4.7343       31.4330        4.5214      -12.8246        3.9866       27.5714
visit4:group1
      30.2239
```

Variance coefficients can be output by specifying the `effects` argument:

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

```
   sigma     k.-1      k.1     k.13
14.47212  0.89592  1.88991  1.35220
```

The first coefficient is the residual standard deviation at the reference timepoint (here -13 week) and the remaining coefficient the residual standard deviation at later timepoints relative to the reference timepoint. It is possible to apply specific transformation on the variance coefficients, for instance to obtain the residual variance at each timepoint:

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

```
sigma.-13  sigma.-1   sigma.1  sigma.13
   14.472    12.966    27.351    19.569
```

4.6 Extract estimated coefficient and associated uncertainty

The uncertainty about the mean coefficients can be obtained using the `model.tables` method [2]:

```
model.tables(eUN.lmm)
```

```
               estimate      se     df     lower    upper    p.value
(Intercept)     38.7290  4.5765 18.003   29.1143 48.34369 1.0891e-07
visit2          -4.7343  2.7759 17.543  -10.5772  1.10851 1.0574e-01
visit3          31.4330  8.6297 17.585   13.2719 49.59411 1.9229e-03
visit4           4.5214  8.0050 17.995  -12.2968 21.33958 5.7917e-01
group1         -12.8246  6.4721 18.003  -26.4219  0.77265 6.3015e-02
visit2:group1    3.9866  3.9957 17.937   -4.4102 12.38329 3.3169e-01
visit3:group1   27.5714 12.4199 17.831    1.4605 53.68232 3.9634e-02
visit4:group1   30.2239 11.3208 17.995    6.4394 54.00840 1.5624e-02
```

Values for the all correlation parameters can be displayed too, by specifying `effect=c("variance","correla`

```
model.tables(eUN.lmm, effect = c("variance","correlation"))
```

```
               estimate       se      df    lower    upper    p.value
sigma        14.4721183 2.412020 15.3158 10.15148 20.63170         NA
k.-1          0.8959206 0.127032 20.2671  0.66670  1.20396 0.44721963
k.1           1.8899095 0.431098 25.9157  1.18244  3.02067 0.00974152
k.13          1.3521979 0.317550 29.8074  0.83694  2.18468 0.20874407
rho(-13,-1)   0.8007214 0.085177 13.4142  0.52949  0.92343 0.00042923
rho(-13,1)    0.2688043 0.219200  7.9286 -0.26374  0.67576 0.27735748
rho(-13,13)  -0.0854578 0.233981  8.5882 -0.55306  0.42309 0.72505145
rho(-1,1)     0.0036838 0.237237  8.1487 -0.49424  0.49979 0.98798445
rho(-1,13)   -0.0941328 0.233649  8.9191 -0.55697  0.41331 0.69821381
rho(1,13)     0.5384239 0.176221 10.2233  0.05058  0.81883 0.03522642
```

Because these parameters are constrained (e.g. strictly positive), they uncertainty is by default computed after transformation (e.g. `log`) and then backtransformed. The column argument can be used to extract more or less information, e.g.:

```
model.tables(eUN.lmm, columns = c("estimate","p.value"))
```

```
               estimate    p.value
(Intercept)     38.7290 1.0891e-07
visit2          -4.7343 1.0574e-01
visit3          31.4330 1.9229e-03
visit4           4.5214 5.7917e-01
group1         -12.8246 6.3015e-02
visit2:group1    3.9866 3.3169e-01
visit3:group1   27.5714 3.9634e-02
visit4:group1   30.2239 1.5624e-02
```

[2] it is equivalent to `confint` method except that by default it also outputs `se` and `p.value`

All parameters can be displayed by specifying `effect="all"`. The functions `add` (resp. `remove`) can be used to add (resp. remove) one or several columns from the default display, e.g.:

```
model.tables(eUN.lmm, columns = add("statistic"))
```

```
               estimate      se statistic     df    lower    upper    p.value
(Intercept)     38.7290  4.5765   8.46260 18.003  29.1143 48.34369 1.0891e-07
visit2          -4.7343  2.7759  -1.70552 17.543 -10.5772  1.10851 1.0574e-01
visit3          31.4330  8.6297   3.64242 17.585  13.2719 49.59411 1.9229e-03
visit4           4.5214  8.0050   0.56482 17.995 -12.2968 21.33958 5.7917e-01
group1         -12.8246  6.4721  -1.98151 18.003 -26.4219  0.77265 6.3015e-02
visit2:group1    3.9866  3.9957   0.99772 17.937  -4.4102 12.38329 3.3169e-01
visit3:group1   27.5714 12.4199   2.21995 17.831   1.4605 53.68232 3.9634e-02
visit4:group1   30.2239 11.3208   2.66977 17.995   6.4394 54.00840 1.5624e-02
```

4.7 Extract estimated residual variance-covariance structure

The method `sigma` can be used to output the modeled residual covariance structure and then converted to a correlation matrix using `cov2cor`:

```
Sigma <- sigma(eUN.lmm)
Sigma
```

```
        -13      -1       1      13
-13 209.442 150.250 106.400 -24.202
-1  150.250 168.114   1.306 -23.884
1   106.400   1.306 748.077 288.184
13  -24.202 -23.884 288.184 382.952
```

```
cov2cor(Sigma)
```

```
       -13     -1     1     13
-13  1.000  0.801 0.269 -0.085
-1   0.801  1.000 0.004 -0.094
1    0.269  0.004 1.000  0.538
13  -0.085 -0.094 0.538  1.000
```

The method can also be used to extract the residual covariance relative to a "known" individual:

```
sigma(eUN.lmm, cluster = 5)
```

```
        -13      1      13
-13 209.442 106.40 -24.202
1   106.400 748.08 288.184
13  -24.202 288.18 382.952
```

or for a new individual:

```
newdata <- data.frame(id = "X", time = c("-13","-1","1","13"))
sigma(eUN.lmm, cluster = newdata)
```

```
        -13       -1        1      13
-13 209.442 150.2502 106.4000 -24.202
-1  150.250 168.1138   1.3064 -23.884
1   106.400   1.3064 748.0769 288.184
13  -24.202 -23.8844 288.1839 382.952
```

4.8 Marginal effects

The `effects` method can be used to evaluate marginal means with respect to a categorical variable:

- $\mathbb{E}\left[Y_t \mid \text{group}\right]$

```
effects(eUN.lmm, variable = "group")
```

```
                Average counterfactual outcome
                 w.r.t 'group' values

                  estimate    se   df  lower   upper
   group=0(t=-13)   38.729 4.576   18 29.114  48.344
   group=0(t=-1)    33.995   4.1 17.9 25.377  42.612
   group=0(t=1)     70.162 8.649 17.7 51.968  88.356
   group=0(t=13)     43.25 6.188   18 30.249  56.251
   group=1(t=-13)   25.904 4.576   18  16.29  35.519
   group=1(t=-1)    25.157 4.167 18.7 16.425  33.889
   group=1(t=1)     84.909 8.951 18.2 66.115 103.702
   group=1(t=13)     60.65 6.188   18 47.649  73.651
```

- $\mathbb{E}\left[Y_t - Y_0 \mid \text{group}\right]$

```
effects(eUN.lmm, type = "change", variable = "group")
```

```
                Average counterfactual change in outcome
                 w.r.t 'group' values

                  estimate    se   df   lower  upper
   group=0(dt=-1)   -4.734 2.776 17.5 -10.577  1.109
   group=0(dt=1)    31.433  8.63 17.6  13.272 49.594
   group=0(dt=13)    4.521 8.005   18 -12.297  21.34
   group=1(dt=-1)   -0.748 2.874 18.3  -6.779  5.283
   group=1(dt=1)    59.004 8.932   18  40.242 77.767
   group=1(dt=13)   34.745 8.005   18  17.927 51.563
```

- $\mathbb{E}\left[\int_0^T Y_t dt \mid \text{group}\right]$

```
effects(eUN.lmm, type = "auc", variable = "group")
```

```
             Average counterfactual area under the outcome curve
              w.r.t 'group' values

             estimate      se   df    lower    upper
group=0(auc) 1220.972 104.098 17.8 1002.072 1439.873
group=1(auc) 1289.782 105.512 18.5 1068.508 1511.056
```

It can also be used to contrast these marginal means:

- $\mathbb{E}\left[Y_t \mid \text{group} = 1\right] - \mathbb{E}\left[Y_t \mid \text{group} = 0\right]$

```
effects(eUN.lmm, type = "difference", variable = "group")
```

```
             Difference in average counterfactual outcome
              w.r.t 'group' values

                  estimate     se   df   lower  upper p.value  
group=1-0(t=-13)   -12.825  6.472   18 -26.422  0.773  0.0630 .
group=1-0(t=-1)     -8.838  5.846 18.3 -21.106   3.43  0.1477  
group=1-0(t=1)      14.747 12.447 17.9 -11.409 40.903  0.2516  
group=1-0(t=13)     17.399  8.752   18  -0.987 35.785  0.0622 .
```

- $\mathbb{E}\left[Y_t - Y_0 \mid \text{group} = 1\right] - \mathbb{E}\left[Y_t - Y_0 \mid \text{group} = 0\right]$

```
effects(eUN.lmm, type = c("change","difference"), variable = "group")
```

```
             Difference in average counterfactual change in outcome
              w.r.t 'group' values

                  estimate     se   df lower  upper p.value  
group=1-0(dt=-1)     3.987  3.996 17.9 -4.41 12.383  0.3317  
group=1-0(dt=1)     27.571  12.42 17.8 1.461 53.682  0.0396 *
group=1-0(dt=13)    30.224 11.321   18 6.439 54.008  0.0156 *
```

- $\mathbb{E}\left[\int_0^T Y_t dt \mid \text{group} = 1\right] - \mathbb{E}\left[\int_0^T Y_t dt \mid \text{group} = 0\right]$

```
effects(eUN.lmm, type = c("auc","difference"), variable = "group")
```

```
             Difference in average counterfactual area under the outcome curve
              w.r.t 'group' values

                estimate     se   df   lower   upper p.value
group=1-0(auc)    68.809 148.22 18.1 -242.44 380.059   0.648
```

It is possible to control the set of covariates used to condition on via the `conditional` argument. This can be useful when considering an interaction with a biomarker to obtain biomarker-specific effects.

4.9 Random effects

Mixed model having a compound symmetry structure with positive correlation parameters may be equivalent to random intercept models, possibly with nested random effects. Indeed in some case the residual variance-covariance matrix can then be decomposed as:

$$\Omega = Z\Psi Z^\intercal + \Delta$$

- Z is the design matrix associated to the random effect (e.g. patient id)
- Ψ is the variance-covariance of the random effects
- Δ the residual variance covariance conditional to the random effects.

One can the use `lme4` syntax to fit random intercept models with `lmm`:

```
eRI.lmm <- lmm(glucagonAUC ~ visit*group + (1|id), data = gastricbypassL)
eRI.lmm
```

```
               Linear Mixed Model with a random intercept

outcome/cluster/time: glucagonAUC/id/XXtimeXX
data                : 78 observations from 20 clusters
parameter           : 8 mean ((Intercept) visit2 visit3 visit4 group1 visit2:group1 visit3:group1 v
                      1 variance (sigma)
                      1 correlation (rho(id))
log-restr.likelihood: -314.394203759159
convergence         : TRUE (6 iterations)
```

It is also possible to specify cross or nested random effects, e.g.:

```
eNRI.lmm <- lmm(glucagonAUC ~ visit*group + (1|id/baseline), data = gastricbypassL)
eNRI.lmm
```

```
               Linear Mixed Model with nested random intercepts

outcome/cluster/time: glucagonAUC/id/XXtimeXX
data                : 78 observations from 20 clusters
parameter           : 8 mean ((Intercept) visit2 visit3 visit4 group1 visit2:group1 visit3:group1 v
                      1 variance (sigma)
                      2 correlation (rho(id/baseline) rho(id))
log-restr.likelihood: -308.994835006264
convergence         : TRUE (6 iterations)
```

We obtain the same log-likelihood as, respectively, `eCS.lmm` and `eBCS.lmm`. Indeed, as previously mentioned, with positive residual correlation the random effect structure is equivalent to a compound symmetry structure.
⚠ random slopes are not currently supported in LMMstar.
⚠ the proposed implementation can be very inefficient compared to `lme4`.

The joint distribution between the outcome $\boldsymbol{Y}$ and the random effects $\boldsymbol{\eta}$ can be expressed as:

$$\begin{bmatrix} \boldsymbol{Y} \\ \boldsymbol{\eta} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \Omega & Z\Psi \\ \Psi Z^\intercal & \Psi \end{bmatrix} \right)$$

Denote by $\varepsilon_i = \boldsymbol{Y}_i - \boldsymbol{\mu}_i$ the vector of marginal residuals relative to individual i, Ω_i its variance-covariance matrix, and $\psi_j = (\Psi)_{jj}$ the variance of the j-th random effect. We can re-express the expected value of the j-th random effect for individual i as:

$$\eta_{ij} = \psi_j Z_{ij} \Omega_i^{-1} \varepsilon_i$$

This is what the `ranef` method returns:

```
head(ranef(eRI.lmm, format = "wide"))
```

```
head(ranef(eNRI.lmm, format = "wide"))
```

```
  id estimate
1  1 -2.51154
2  2  1.01043
3  3  6.08384
4  4 -6.62350
5  5  0.39519
6  6 -2.73384
```

```
  id  estimate estimate.FALSE estimate.TRUE
1  1 -0.494271       -3.50959      -3.23209
2  2  0.186051      -10.39431      12.93198
3  3  1.088409        9.36327       5.48225
4  4 -1.219596      -11.06703      -5.56784
5  5  0.081686       -0.71254       1.82672
6  6 -0.503386       -7.81700       0.95098
```

It is also possible to extract the variance decomposition by setting the argument `effects` to `"variance"`:

```
ranef(eRI.lmm, effects = "variance",
      format = "wide")
```

```
ranef(eNRI.lmm, effects = "variance",
      format = "wide")
```

```
      type absolute relative
1    total  380.580  1.00000
2       id   82.741  0.21741
3 residual  297.839  0.78259
```

```
      type absolute relative
1    total  380.957 1.000000
2       id   15.465 0.040595
3 baseline  210.938 0.553705
4 residual  154.554 0.405700
```

Confidence intervals can also be obtained setting the argument `se` to `TRUE` and `format` equal to `"long"`:

```
head(ranef(eRI.lmm, se = TRUE))
```

```
  id estimate     se      df    lower   upper
1  1 -2.51154 2.3019 11.1302  -7.5708  2.5477
2  2  1.01043 2.1163 15.7355  -3.4821  5.5030
3  3  6.08384 2.9771  6.2085  -1.1421 13.3098
4  4 -6.62350 3.1114  5.8319 -14.2902  1.0432
5  5  0.39519 1.9661 23.8446  -3.6640  4.4543
6  6 -2.73384 2.2940 10.0189  -7.8438  2.3761
```

4.10 Sum of squares

⚠ The definition of the sum of squares is not straightforward with mixed models. Intuitively summing residuals across several outcomes will be hard to interpret unless all outcomes have the same variance. This is why LMMstar does not provide them. Nevertheless for specific covariance structure, namely independence and compound symmetry (with positive correlation) structure, sum of squares can be deduced from the `lmm` object - see appendix C for the theoretical derivations. Importantly, with these structures the residuals can be reparametrised as random effects plus independent residuals, i.e. $\Omega = Z\Psi Z^\intercal + \delta I$ where I is the identity matrix and δ the variance of these independent residuals.

Appendix C illustrate how to extract the sum of squares for univariate linear regression (i.e. independence structure) and here we illustrate the case of a compound symmetry structure. A key step is to extract from the `lmm` object the conditional residual variance δ:

```
sigma2 <- coef(eCS.lmm, effect = "variance")^2
tau <- coef(eCS.lmm, effect = "correlation")*sigma2
delta <- unname(sigma2 - tau)
```

This step will typically depend on the covariance structure. The residual sum of squares (SSE) equals the residual degrees of freedom times the conditional variance:

```
df.res <- df.residual(eCS.lmm)
SSE <- df.res * delta
c(df.res = df.res, SSE = SSE)
```

```
df.res    SSE
    70  20849
```

For the regression sum of squares (SSR), we first extract the mean parameters and their variance-covariance based on the expected information:

```
eBeta.lmm <- coef(eCS.lmm)
eVcov.lmm <- vcov(eCS.lmm, type.information = "expected")
```

Parameters are grouped with respect to the original variable:

```
attr(model.matrix(eCS.lmm),"assign")
```

```
[1] 0 1 1 1 2 3 3 3
```

So we respect this grouping when computing the normalized SSR:

```
SSRstar.time <- eBeta.lmm[2:4] %*% solve(eVcov.lmm[2:4,2:4]) %*% eBeta.lmm[2:4]
SSRstar.group <- eBeta.lmm[5] %*% solve(eVcov.lmm[5,5]) %*% eBeta.lmm[5]
```

The SSR is obtained by multiplying the normalized SSR by the conditional variance:

```
SSR.time <- as.double(SSRstar.time * delta)
SSR.group <- as.double(SSRstar.group * delta)
c(time = SSR.time, group = SSR.group)
```

```
   time   group
7872.19  643.57
```

4.11 Proportion of explained variance and partial correlation

For a univariate linear model with homoschedastic residual variance, the proportion of explained variance, also called partial R^2 or partial η^2, is defined as the ratio between sum of squares (e.g. Lakens (2013), equation 12):

$$R^2 = \frac{SSR}{SSR + SSE}$$

```
c(SSR.time/ (SSR.time + SSE),
  SSR.group/ (SSR.group + SSE))
```

```
[1] 0.274092 0.029944
```

Computing the SSR for each individual coefficients, taking its squared root, and multiplying by the sign of the corresponding coefficient leads to the partial correlation. This procedure extends to covariance structures that can be reparametrised as random effects plus independent residuals (see previous subsection) such as the compound symmetry with non-negative correlation.

⚠ for other covariance structures, especially when the variance may be repetition-dependent, the definition of explained variance/partial correlation is not straightforward.

```
eCS.R2 <- partialCor(eCS.lmm, R2 = TRUE)
summary(eCS.R2)
```

```

               Partial correlation

                estimate    se   df  lower upper p.value
  visit2          -0.073 0.119 52.4 -0.311 0.165 0.54028
  visit3           0.438 0.089 51.4   0.26 0.616 < 1e-04
  visit4            0.07 0.119 52.4 -0.168 0.308 0.55876
  group1          -0.173 0.114 60.7 -0.402 0.056 0.13527
  visit2:group1    0.041 0.119 52.8 -0.198  0.28 0.73256
  visit3:group1    0.284 0.106   52  0.071 0.497 0.01007
  visit4:group1    0.314 0.103   52  0.107 0.521 0.00365
  ------------------------------------------------------
 Columns lower and upper contain 95% pointwise confidence intervals for each coefficient.
 Degrees of freedom were computed using a Satterthwaite approximation (column df).

               Coefficient of determination (R2)

              estimate    se   df  lower upper p.value
  visit          0.274  0.08 50.5  0.114 0.434  0.0012
  group           0.03  0.04 60.7 -0.049 0.109  0.4520
  visit:group    0.147 0.073 51.7 <0.001 0.295  0.0500
  global         0.598 0.053 40.4  0.492 0.705  <1e-04
  ----------------------------------------------------
 Columns lower and upper contain 95% pointwise confidence intervals for each coefficient.
 Degrees of freedom were computed using a Satterthwaite approximation (column df).
```

Here the line "global" refer to the R2 for all covariates, computed based on the SSR relative to all mean parameters but the intercept.

⚠ `partialCor` will compute values for all types of mixed models. But their interpretation as partial correlation and proportion of explained variance outside the compound symmetry with non-negative correlation is questionnable.

Note: Other software packages like `effectsize::eta_squared` uses another formula to estimate the partial R2:

$$R^2 = \frac{F df_{num}}{F df_{num} + df_{denom}}$$

where F denote the F-statistic, df_{num} (resp. df_{denom}) the degrees of freedom of the numerator (resp. denominator) of this statistic. However since the calculation of degrees of freedom in LMM is approximate, I would expect this approach to be less reliable than the one of `partialCor` based on the SSR and SSE.

```
aCS.aov <- anova(eCS.lmm)$multivariate
setNames(with(aCS.aov, statistic*df.num/(statistic*df.num+df.denom)), aCS.aov$test)
```

```
   visit       group visit:group 
0.335374    0.033811    0.186290 
```

4.12 Model diagnostic

The method `residuals` returns the residuals in the wide format:

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

```
  id    r.-13     r.-1       r.1     r.13
1  1 -0.36029 -0.11344  0.377177 -1.45539
2  2  0.77339  2.12301 -0.232908 -0.10708
3  3  1.14219 -1.44778 -0.654876  2.01259
4  4 -0.77473  0.20612 -0.127117 -1.39519
5  5  0.22435       NA  0.011432 -0.15398
6  6  0.27439 -0.67308 -1.031131  0.42724
```

or in the long format:

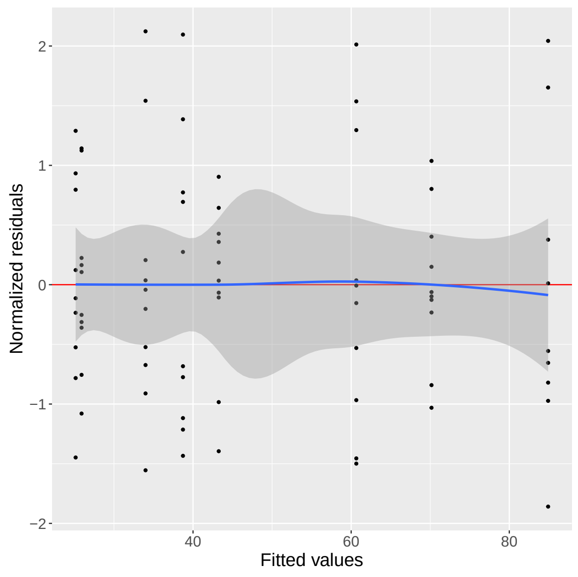
```
eUN.diagL <- residuals(eUN.lmm, type = "normalized", format = "long", keep.data = TRUE)
head(eUN.diagL)
```

```
  id visit time weight glucagonAUC group baseline fitted r.normalized
1  1     1  -13  127.2      20.690     1     TRUE 25.904     -0.36029
2  2     1  -13  165.2      49.922     0     TRUE 38.729      0.77339
3  3     1  -13  109.7      42.434     1     TRUE 25.904      1.14219
4  4     1  -13  146.2      27.517     0     TRUE 38.729     -0.77473
5  5     1  -13  113.1      29.151     1     TRUE 25.904      0.22435
6  6     1  -13  158.8      42.700     0     TRUE 38.729      0.27439
```

Various type of residuals can be extract but the normalized one are recommanded when doing model checking. Diagnostic plots can then be generated by the user, or directly from the `lmm` object via the method `plot` (which internally calls the `residuals` method):

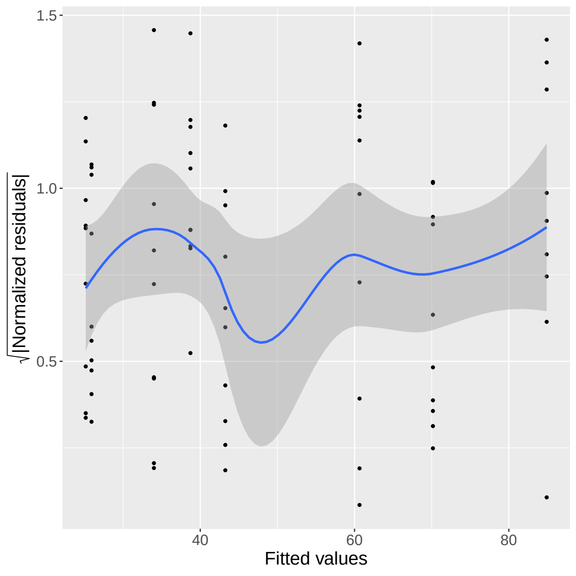
- misspecification of the mean structure

```
plot(eUN.lmm, type = "scatterplot")
```



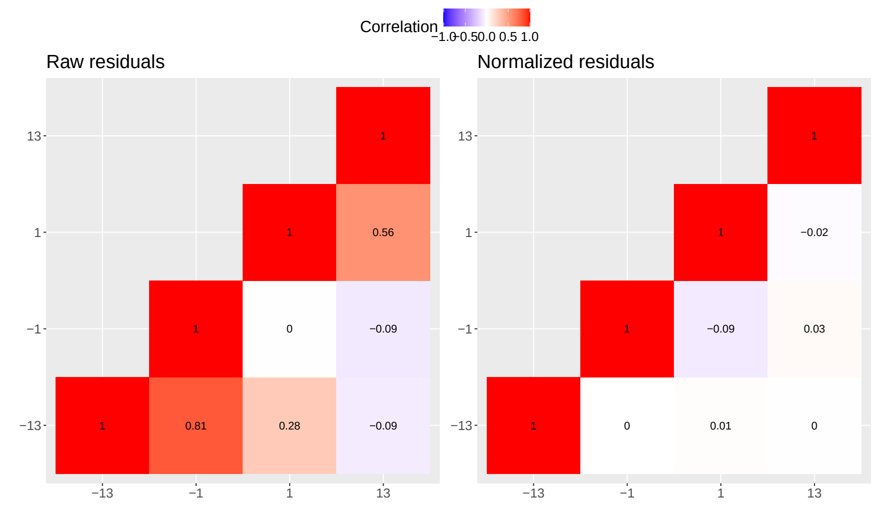
- misspecification of the variance structure

```
plot(eUN.lmm, type = "scatterplot2")
```



- misspecification of the correlation structure

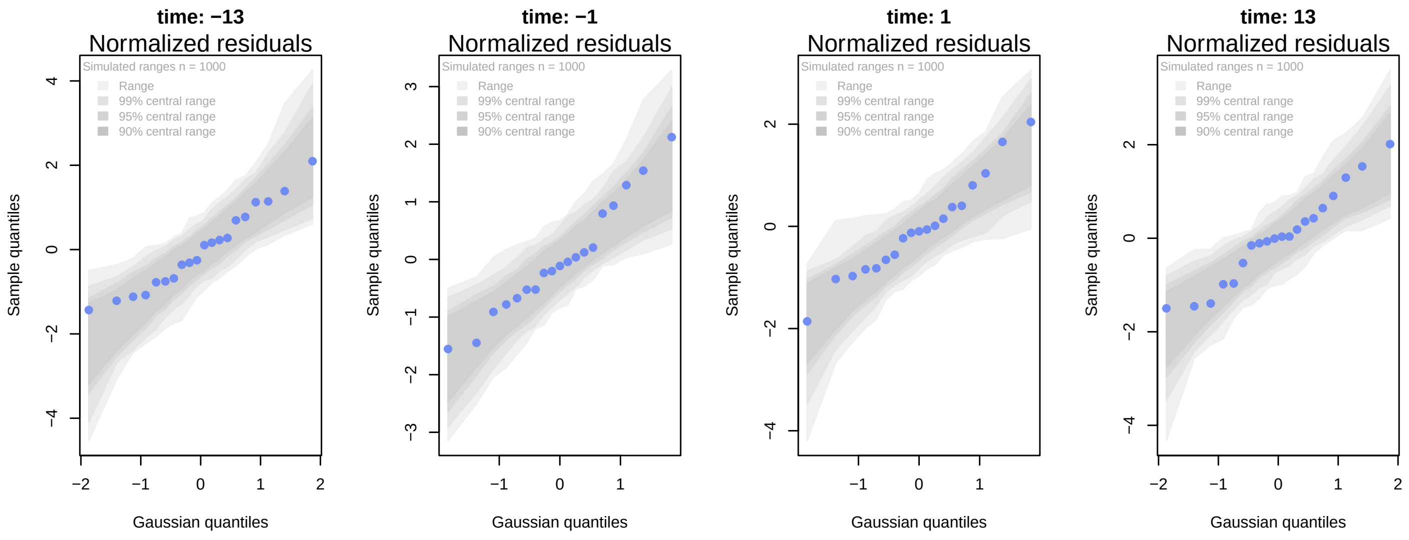
```
plot(eUN.lmm, type = "correlation", type.residual = "response")
plot(eUN.lmm, type = "correlation", type.residual = "normalized")
```



- residual distribution vs. normal distribution [3]:

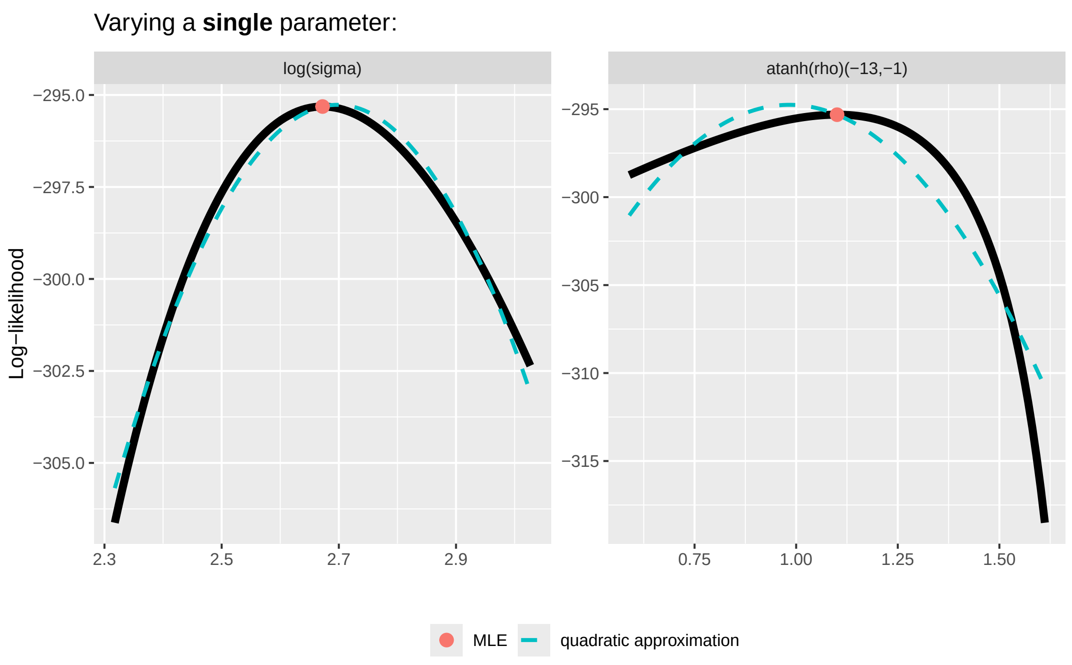
```
plot(eUN.lmm, type = "qqplot", engine = "qqtest",
     facet = ~time, labeller = "label_both", facet_nrow=1)
## Note: the qqtest package to be installed to use the argument engine.plot = "qqtest"
```

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



⚠ Deviation from the normal distribution does not necessarily question the validity of the statistical inference. Moreover, for variance and correlation parameters, normally distributed data is not enougth to ensure valid statistical inference. Instead one could assess whether the log-likelihood is locally quadratic as this ensures normally distributed estimates in finite samples (Geyer, 2013). Since the likelihood function is a multi-dimensional function this is not an easy task but one can look at specific 'slices' using the `profile` method:

```
eUN.lmm_profile <- profile(eUN.lmm, effects = c("sigma","rho(-13,-1)"))
plot(eUN.lmm_profile)
```



4.13 Visualize model fit

The fitted values can be displayed via the `plot` method:

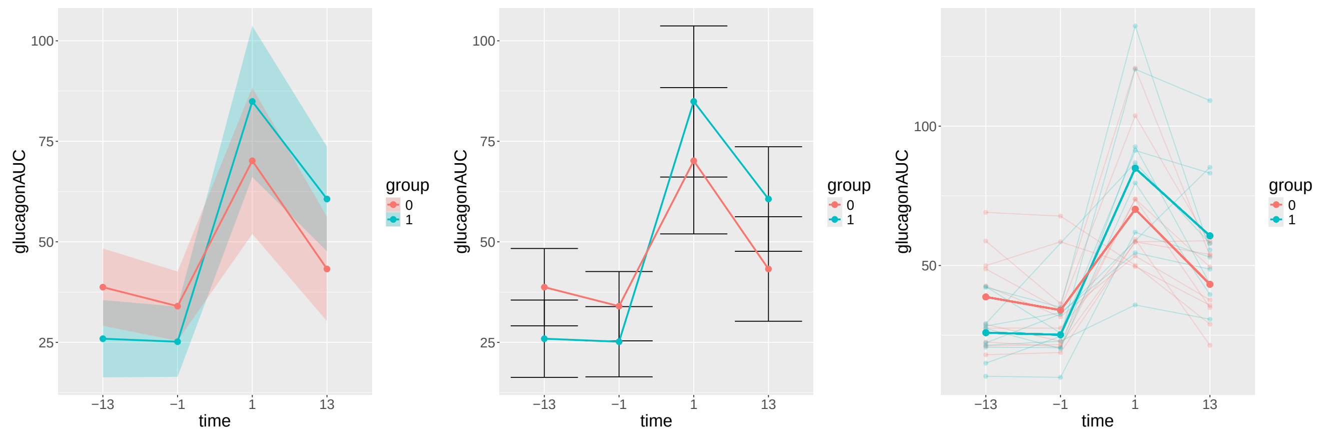
```
## left panel
plot(eUN.lmm, type = "fit", color = "group", size.text = 20)
```

⚠ the shaded area represent 95% confidence intervals (CIs), i.e. is not adjusted for multiplicity over time. More explicit (but sometimes less readable) representation of the CIs can be obtained by setting the argument `ci.alpha` to `NA`:

```
## middle panel
plot(eUN.lmm, type = "fit", color = "group", ci.alpha = NA, size.text = 20)
```

It is also possible to display the observed values along with the fitted values by setting the argument `obs.alpha` to a strictly positive value below or equal to 1. This argument controls the transparency of the color used to display the observed values:

```
## right panel
plot(eUN.lmm, type = "fit", obs.alpha = 0.25, ci = FALSE, size.text = 20)
```



When considering continuous covariates, e.g.:

```
## add baseline weight
gastricbypassLB <- merge(gastricbypassL, gastricbypassW[c("id","weight1")], by = "id")

eUN.lmmB <- lmm(glucagonAUC ~ weight1 + visit*group, repetition = ~time|id,
                structure = "UN", data = gastricbypassLB)
```

The default graphical display can be confusing as it shows one curve per distinct set of covariate values:

```
## left panel
plot(eUN.lmmB, type = "fit", color = "group", ci = FALSE, size.text = 20)
```

However it is possible to restrict the display specific to a covariate value via the argument `at`:

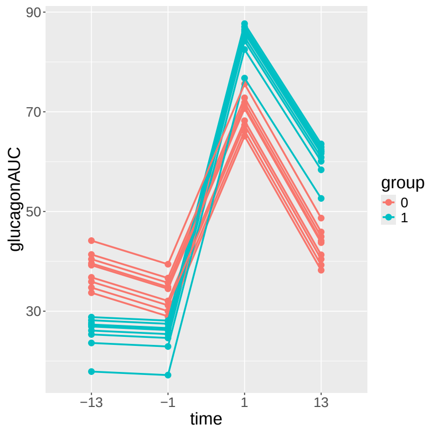
```
## middel panel
plot(eUN.lmmB, type = "fit", color = "group", ci = FALSE, size.text = 20,
     at = data.frame(weight1 = 150))
```

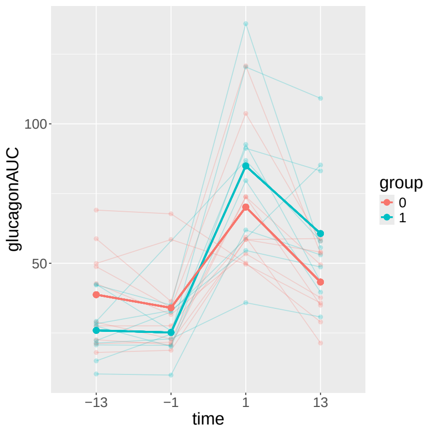
The `plot` method calls the `autoplot` methods which returns a list containing:

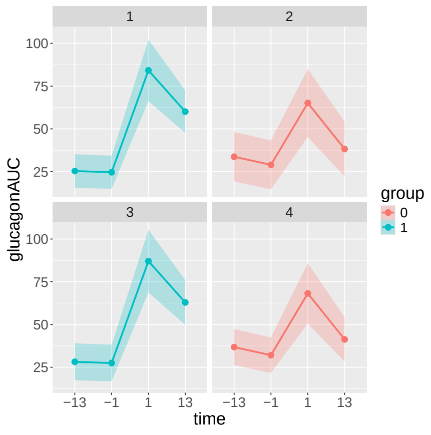
- a ggplot2 object (element `plot`)
- the dataset used to generate the ggplot2 object (element `data`)

This should ease further customization of the graphical display, e.g.:

```
## right panel
gg.traj <- autoplot(eUN.lmmB, type = "fit", color = "group", size.text = 20, facet =~id)
gg.traj$plot %+% gg.traj$data[gg.traj$data$id %in% 1:4,]
```







4.14 Partial residuals

In a linear model where we split the covariates and mean parameters into two sets:

$$Y_i = X_{1,i}\beta_1 + X_{2,i}\beta_2 + \varepsilon_i$$

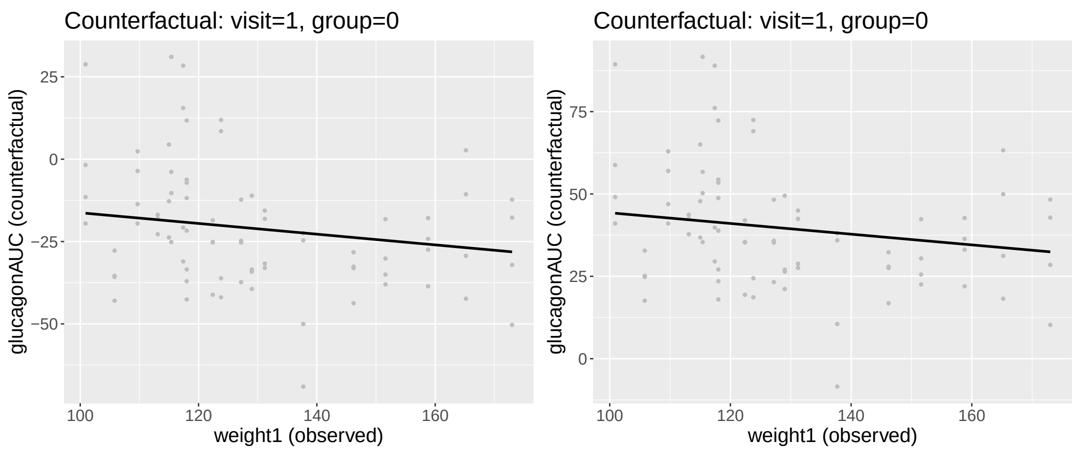
the partial residuals w.r.t. to the covariate(s) X_2 are defined by $\varepsilon_i^{X_2} = Y_i - X_{1,i}\beta_1$.
They can be computed via the `residuals` method:

```
df.pres <- residuals(eUN.lmmB, type = "partial", variable = "weight1", keep.data = TRUE)
head(df.pres)
```

```
  id visit time weight glucagonAUC group baseline weight1  fitted r.partial
1  1     1  -13  127.2      20.690     0     TRUE   127.2 -20.684  -25.3242
2  1     1    1  115.5      92.600     0    FALSE   127.2 -20.684  -12.2923
3  1     1   -1  120.7      20.535     0     TRUE   127.2 -20.684  -24.7703
4  1     1   13  108.1      43.434     0    FALSE   127.2 -20.684  -37.3259
5 10     1   13   90.9      57.942     0    FALSE   118.0 -19.188   -7.1423
6 10     1    1   99.3     103.728     0    FALSE   118.0 -19.188   11.7323
```

In the output, the X_1 covariates (`time` and `group`) have been set to the reference level (`-13` and `0`) for all observations. Confusion with the ordering variable from the `repetition` argument of `lmm` was avoided by using a different 'time' variable in the mean (`time`) and repetition argument (`visit`) when calling `lmm`. These residuals can be directly displayed via the `plot` method:

```
## left panel
plot(eUN.lmmB, type = "partial", variable = "weight1")
## right panel
plot(eUN.lmmB, type = "partial", variable = c("(Intercept)","weight1"))
```



The `plot` methods can handle one continuous and one categorical covariate (in addition to the intercept) to display interaction plots. In that case each observation/fitted line is colored according to the categorical covariate.

4.15 Statistical inference (linear)

The `anova` method can be use to test one or several linear combinations of the model coefficients using Wald tests. By default, it will simultaneously test all parameters associated to a variable:

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

```
             Multivariate Wald test

               F-statistic        df  p.value
mean: time          86.743 (3,19.0) 2.84e-11 ***
    : glucagon      13.518 (1,13.7)  0.00257  **
```

Note that here the p-values are not adjust for multiple comparisons over variables. It is possible to specify a null hypothesis to be test: e.g. is there a change in average weight just after taking the treatment:

```
anova(eUN.lmm, effects = c("timeA1w-timeB1w=0"))
```

```
             Multivariate Wald test

       F-statistic        df  p.value
all: 1      43.141 (1,17.9) 3.72e-06 ***
```

One can also simulateneously tests several null hypotheses:

```
e.anova <- anova(eUN.lmm, effects = c("timeA1w-timeB1w=0","timeA3m-timeB1w=0"))
summary(e.anova)
```

```
                Multivariate Wald test

          F-statistic        df  p.value
   all: 1      98.651 (2,18.6) 1.23e-10 ***
   ----------------------------------------
  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1.
  Degrees of freedom were computed using a Satterthwaite approximation (column df).

                Univariate Wald test

                     estimate    se   df   lower   upper p.value
   timeA1w - timeB1w   -3.906 0.595 17.9  -5.325  -2.487   3e-05 ***
   timeA3m - timeB1w   -18.24 1.323   19 -21.397 -15.083  <1e-05 ***
   ---------------------------------------------------------------
  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1.
  Columns lower/upper/p.value adjusted for multiple comparisons -- max-test.
  (1e+05 samples have been used)
  Model-based standard errors are derived from the observed information (column se).
  Degrees of freedom were computed using a Satterthwaite approximation (column df).
```

or return all pairwise comparisons for a given factor using the `mcp` function of the multcomp package:

```
library(multcomp)
summary(anova(eUN.lmm, effects = mcp(time = "Tukey")))
```

```
Singular contrast matrix: contrasts "A1w - B1w" "A3m - B1w" "A3m - A1w" have been removed.

                Multivariate Wald test

             F-statistic       df  p.value
   all: time      86.743 (3,19.0) 2.84e-11 ***
   -------------------------------------------
  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1.
  Degrees of freedom were computed using a Satterthwaite approximation (column df).

                Univariate Wald test

             estimate    se   df   lower   upper p.value
   B1w - B3m   -7.882 0.713 19.2  -9.817  -5.947  <1e-05 ***
   A1w - B3m  -11.788 1.018 21.6 -14.549  -9.027  <1e-05 ***
   A3m - B3m  -26.122 1.656 18.8 -30.617 -21.628  <1e-05 ***
   A1w - B1w   -3.906 0.595 17.9  -5.519  -2.292  <1e-05 ***
   A3m - B1w   -18.24 1.323   19 -21.829 -14.651  <1e-05 ***
   A3m - A1w  -14.334 1.057 20.3 -17.201 -11.468  <1e-05 ***
   ---------------------------------------------------------
  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1.
  Columns lower/upper/p.value adjusted for multiple comparisons -- max-test.
  (1e+05 samples have been used)
  Model-based standard errors are derived from the observed information (column se).
  Degrees of freedom were computed using a Satterthwaite approximation (column df).
```

Here the `summary` method prints not only the global test but also the result associated to each hypothesis. When testing transformed variance or correlation parameters, parentheses (as in `log(k).B1w`) cause problem for recognizing parameters:

```
try(
  anova(eUN.lmm,
        effects = c("log(k).B1w=0","log(k).A1w=0","log(k).A3m=0"))
)
```

```
Error in .anova_Wald(object, effects = effects, robust = robust, rhs = rhs,  :
  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).B1w
          ^
```

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

```
name.coef <- rownames(confint(eUN.lmm, effects = "all"))
name.varcoef <- grep("^k",name.coef, value = TRUE)
C <- matrix(0, nrow = 3, ncol = length(name.coef), dimnames = list(name.varcoef, name.coef))
diag(C[name.varcoef,name.varcoef]) <- 1
C[,1:9]
```

```
      (Intercept) timeB1w timeA1w timeA3m glucagon sigma k.B1w k.A1w k.A3m
k.B1w           0       0       0       0        0     0     1     0     0
k.A1w           0       0       0       0        0     0     0     1     0
k.A3m           0       0       0       0        0     0     0     0     1
```

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

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

```
             Multivariate Wald test

       F-statistic       df p.value
all: 1       6.203 (3,18.0) 0.00442 **
```

Note that using the approach of Pipper et al. (2012) it is also possible to adjust for multiple testing across several `lmm` objects. To do so, one first fit the mixed models, then use the **anova** method to indicate which hypotheses are being tested, and combine them using `rbind`. Here is an (artificial) example:

```
Manova <- rbind(anova(eInd.lmm, effects = "glucagon = 0", robust = FALSE),
                anova(eCS.lmm, effects = "glucagon = 0", robust = FALSE),
                anova(eUN.lmm, effects = "glucagon = 0", robust = FALSE),
                name = c("Ind","CS","UN"))
summary(Manova)
```

```
                Multivariate Wald test

          Chi2-statistic      df  p.value
   all: 1          8.893 (3,Inf) 6.88e-06 ***
   ------------------------------------------
  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1.

                Univariate Wald test

                 estimate    se   df   lower  upper  p.value
   Ind: glucagon    -8.27 2.579 34.2 -14.414 -2.126 0.003988  **
   CS: glucagon     0.822  0.62 53.8  -0.655  2.299 0.450012
   UN: glucagon    -0.888 0.242 13.7  -1.464 -0.313 0.000711 ***
   ------------------------------------------------------------
  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1.
  Columns lower/upper/p.value adjusted for multiple comparisons -- max-test.
  (error when computing the adjusted columns lower/upper/p.value by numerical integration: 1.52e-05
  Model-based standard errors are derived from the observed information (column se).
```

4.16 Statistical inference (non-linear)

The `estimate` function can be used to test one or several non-linear combinations of model coefficients, using a first order delta method to quantify uncertainty. The combination has to be specified via a function (argument `f`). To illustrate its use consider an ANCOVA analysis:

$$Y_{i1} = \alpha + \beta Y_{i,0} + \gamma X_i + e_i$$

```
e.ANCOVA <- lm(weight4 ~ weight1 + group, data = gastricbypassW)
summary(e.ANCOVA)$coef
```

```
              Estimate Std. Error    t value     Pr(>|t|)
(Intercept) -5.9285136 8.78006389 -0.6752244 5.086140e-01
weight1      0.8236279 0.06411563 12.8459772 3.524665e-10
group        4.1404554 2.53335466  1.6343765 1.205604e-01
```

We can replicate this analysis by first fitting a mixed model:

$$Y_{ij} = \alpha_j + \gamma_j X_i + \varepsilon_{i,j} \text{ where } \varepsilon_i \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}\right)$$

```
dfL14 <- dfL[dfL$visit %in% c(1,4),]
dfL14$time <- droplevels(dfL14$time)
e.lmmANCOVA <- lmm(weight ~ time+time:group, repetition = ~time|id,
                   data = dfL14)
```

and then perform a first order delta-method:

```
lava::estimate(e.lmmANCOVA, f = function(p){
  c(Y1 = as.double(p["rho(B3m,A3m)"]*p["k.A3m"]),
    X1 = as.double(p["timeA3m:group"]-p["rho(B3m,A3m)"]*p["k.A3m"]*p["timeB3m:group"]))
})
```

```
    estimate         se        df      lower     upper      p.value
Y1 0.8236279 0.06230919  9.874633  0.6845551 0.9627007 1.332743e-07
X1 4.1404554 2.46197819 15.161269 -1.1022695 9.3831803 1.130927e-01
```

Indeed:

$$\begin{aligned} \mathbb{E}\left[Y_{i2}|Y_{i1}, X_i\right] &= \alpha_2 + \gamma_2 X_i + \rho\frac{\sigma_2}{\sigma_1}\left(Y_{i1} - \alpha_1 - \gamma_1 X_i\right) \\ &= \alpha_2 - \rho\frac{\sigma_2}{\sigma_1}\alpha_1 + \rho\frac{\sigma_2}{\sigma_1}Y_{i1} + \left(\gamma_2 - \rho\frac{\sigma_2}{\sigma_1}\gamma_1\right) X_i \end{aligned}$$

We obtain identical estimate but different standard-errors/degrees of freedom compared to the univariate linear model approach. The later is to be prefer as it does not rely on approximation. The former is nevertheless useful as it can handle missing data in the outcome variable.

4.17 Baseline adjustment

In clinical trial the group and intervention variable often do not coincide, e.g., in presence of baseline measurement. In our running example, the first two measurement are pre-treatment (i.e. treatment should be "none") while the last two measurements are post-treatment (i.e. treatment should be 1 or 2). The baselineAdjustment function can be helpful to:

- define the treatment variable from the time and allocation variable, where baseline has its specific value

```
gastricbypassL$treat <- baselineAdjustment(gastricbypassL, variable = "group",
                                repetition = ~time|id, constrain = c("B3m","B1w"),
                                new.level = "none")
table(treat = gastricbypassL$treat, time = gastricbypassL$time, group = gastricbypassL$group)
```

```
, , group = 0

      time
treat   B3m B1w A1w A3m
  none   10  10   0   0
  0       0   0  10  10
  1       0   0   0   0

, , group = 1

      time
treat   B3m B1w A1w A3m
  none   10  10   0   0
  0       0   0   0   0
  1       0   0  10  10
```

- define the treatment variable from the time and allocation variable, where baseline corresponds to the reference group

```
gastricbypassL$treat2 <- baselineAdjustment(gastricbypassL, variable = "group",
                                            repetition = ~time|id, constrain = c("B3m","B1w")
   )
table(treat = gastricbypassL$treat2, time = gastricbypassL$time, group = gastricbypassL$group)
```

```
, , group = 0

     time
treat B3m B1w A1w A3m
    1  10  10   0   0
    0   0   0  10  10
```

```
, , group = 1

     time
treat B3m B1w A1w A3m
    1  10  10  10  10
    0   0   0   0   0
```

- define a time varying treatment variable from the time and allocation variable

```
gastricbypassL$timeXtreat <- baselineAdjustment(gastricbypassL, variable = "group",
                                                repetition = ~time|id, constrain = c("B3m","
   B1w"),
                                                collapse.time = ".")

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

```
, , group = 0

       time
treat   B3m B1w A1w A3m
  B3m    10   0   0   0
  B1w     0  10   0   0
  A1w.0   0   0  10   0
  A3m.0   0   0   0  10
  A1w.1   0   0   0   0
  A3m.1   0   0   0   0

, , group = 1

       time
treat   B3m B1w A1w A3m
  B3m    10   0   0   0
  B1w     0  10   0   0
  A1w.0   0   0   0   0
  A3m.0   0   0   0   0
  A1w.1   0   0  10   0
  A3m.1   0   0   0  10
```

We would then typically like to model group differences only after baseline (i.e. only at 1 week and 3 months after). This can be performed using the time varying treatment variable, e.g.:

```
eC.lmm <- lmm(weight ~ timeXtreat, data = gastricbypassL,
              repetition = ~time|id, structure = "UN")
coef(eC.lmm) ## change from baseline
```

```
(Intercept)   timeXtreatB1w timeXtreatA1w.0 timeXtreatA3m.0 timeXtreatA1w.1 timeXtreatA3m.1
  128.97000        -7.73000       -13.38978       -28.52130       -13.15022       -24.68870
```

or

```
eC2.lmm <- lmm(weight ~ 0 + timeXtreat, data = gastricbypassL,
               repetition = ~time|id, structure = "UN")
coef(eC2.lmm) ## absolute value
```

```
timeXtreatB3m   timeXtreatB1w timeXtreatA1w.0 timeXtreatA3m.0 timeXtreatA1w.1 timeXtreatA3m.1
     128.9700        121.2400        115.5802        100.4487        115.8198        104.2813
```

The parametrization however does not (directly) output treatment effects. Instead one may be tempted to use a formula like `treatment*time`. However this will lead to a non-indentifiable model. Indeed we are only able to estimate a total of 6 means when constraining the expected baseline value between the two groups to be the same. 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)"    "treat0"         "treat1"         "timeB1w"        "timeA1w"
 [6] "timeA3m"        "treat0:timeB1w" "treat1:timeB1w" "treat0:timeA1w" "treat1:timeA1w"
[11] "treat0:timeA3m" "treat1:timeA3m"
```

contains 12 parameters (i.e. 6 too many). Fortunately, the `lmm` will drop non-identifiable effects from the model and fit the resulting simplified model:

```
eC3.lmm <- lmm(weight ~ treat2*time, data = gastricbypassL,
                repetition = ~time|id, structure = "UN")
```

```
Constant values in the design matrix for the mean structure.
Coefficients "treat20" "treat20:timeB1w" relative to interactions "treat2:time" have been removed.
```

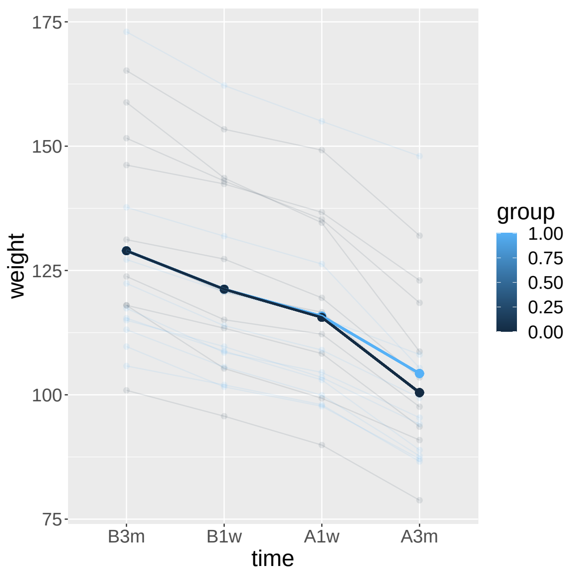
with the following coefficients:

```
model.tables(eC3.lmm)
```

```
                  estimate        se       df      lower       upper      p.value
(Intercept)     128.9700000 4.5323695 18.98130 119.483009 138.4569912 0.000000e+00
timeB1w          -7.7300000 0.6974427 18.97552  -9.189892  -6.2701082 9.938186e-10
timeA1w         -13.1502219 0.8970429 22.87334 -15.006465 -11.2939786 4.058975e-13
timeA3m         -24.6886957 1.7751662 22.25061 -28.367762 -21.0096290 1.863398e-12
treat20:timeA1w  -0.2395562 0.6484895 17.66860  -1.603816   1.1247037 7.162149e-01
treat20:timeA3m  -3.8326086 2.1066817 17.60613  -8.265691   0.6004734 8.592047e-02
```

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

```
plot(eC3.lmm, color = "group", ci = FALSE, size.text = 20, obs.alpha = 0.1)
```



4.18 Marginal means

The `emmeans` package can be used to output marginal means. Consider the following model:

```
dfL$group2 <- as.numeric(dfL$id) %% 3 == 0
e.group <- lmm(glucagon ~ time*group2, data = dfL,
               repetition = ~time|id, structure = "UN")
```

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

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

```
NOTE: Results may be misleading due to involvement in interactions
 time emmean    SE   df lower.CL upper.CL
 B3m    4.45 0.156 18.0     4.12     4.78
 B1w    4.32 0.131 18.0     4.05     4.60
 A1w    5.95 0.262 18.4     5.40     6.50
 A3m    5.12 0.187 18.0     4.73     5.51

Results are averaged over the levels of: group2
Confidence level used: 0.95
```

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

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

```
  time observed missing     mean        sd      min       q1   median       q3      max
1  B3m       20       0 4.514352 0.1502565 4.290643 4.290643 4.610227 4.610227 4.610227
2  B1w       19       0 4.384638 0.1643256 4.149209 4.149209 4.493298 4.493298 4.493298
3  A1w       19       0 6.060587 0.2030012 5.729961 5.954314 6.178668 6.178668 6.178668
4  A3m       20       0 5.057642 0.1465315 4.964144 4.964144 4.964144 5.275805 5.275805
```

as the groups are not balanced:

```
table(group = dfL$group2, time = dfL$time)
```

```
       time
group   B3m B1w A1w A3m
  FALSE  14  13  14  14
  TRUE    6   6   5   6
```

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
```

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 = ~group2|time)
emmeans.group
```

```
time = B3m:
 group2 emmean    SE   df lower.CL upper.CL
 FALSE    4.61 0.171 18.0     4.25     4.97
  TRUE    4.29 0.262 18.0     3.74     4.84

time = B1w:
 group2 emmean    SE   df lower.CL upper.CL
 FALSE    4.49 0.145 18.4     4.19     4.80
  TRUE    4.15 0.219 17.9     3.69     4.61

time = A1w:
 group2 emmean    SE   df lower.CL upper.CL
 FALSE    6.18 0.277 17.8     5.60     6.76
  TRUE    5.73 0.446 18.6     4.80     6.66

time = A3m:
 group2 emmean    SE   df lower.CL upper.CL
 FALSE    4.96 0.205 18.0     4.53     5.39
  TRUE    5.28 0.313 18.0     4.62     5.93

Confidence level used: 0.95
```

Using the pair function displays the differences:

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

```
time = B3m:
 contrast     estimate    SE   df t.ratio p.value
 TRUE - FALSE   -0.320 0.313 18.0  -1.022  0.3202

time = B1w:
 contrast     estimate    SE   df t.ratio p.value
 TRUE - FALSE   -0.344 0.262 18.0  -1.311  0.2062

time = A1w:
 contrast     estimate    SE   df t.ratio p.value
 TRUE - FALSE   -0.449 0.525 18.4  -0.855  0.4034

time = A3m:
 contrast     estimate    SE   df t.ratio p.value
 TRUE - FALSE    0.312 0.374 18.0   0.834  0.4153
```

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
 TRUE - FALSE B3m    -0.320 0.313 18.0   -1.156    0.517  -1.022  0.6926
 TRUE - FALSE B1w    -0.344 0.262 18.0   -1.046    0.358  -1.311  0.5061
 TRUE - FALSE A1w    -0.449 0.525 18.4   -1.853    0.955  -0.855  0.7960
 TRUE - FALSE A3m     0.312 0.374 18.0   -0.689    1.312   0.834  0.8084

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(eC3.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
 treat20 - treat21 B3m      0.00 0.000 Inf     NaN     NaN
 treat20 - treat21 B1w      0.00 0.000 Inf     NaN     NaN
 treat20 - treat21 A1w     -0.24 0.648  18  -0.369  0.9195
 treat20 - treat21 A3m     -3.83 2.107  18  -1.819  0.1645

P value adjustment: sidak method for 2 tests
```

4.19 Predictions

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

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

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

```
  estimate       se       df     lower    upper
1 132.9801 4.664247 19.75815 123.24305 142.7172
2 125.0979 4.388294 19.91418 115.94155 134.2543
3 121.1922 4.214230 20.55331 112.41660 129.9678
4 106.8577 3.942058 20.95499  98.65871 115.0568
```

which can be computing by creating a design matrix:

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

```
   (Intercept) timeB1w timeA1w timeA3m 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(eUN.lmm)
```

```
       [,1]
1  132.9801
21 125.0979
41 121.1922
61 106.8577
```

- **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 = "B3m", weight = coef(eUN.lmm)["(Intercept)"], glucagon = 0),
  data.frame(id = 1, time = "B1w", weight = NA, glucagon = 0),
  data.frame(id = 2, time = "B3m", weight = 100, glucagon = 0),
  data.frame(id = 2, time = "B1w", weight = NA, glucagon = 0)
)
predict(eUN.lmm, newdata = newd, type = "dynamic", keep.newdata = TRUE)
```

```
  id time   weight glucagon  estimate        se  df     lower    upper
1  1  B3m 132.9801        0        NA        NA Inf        NA       NA
2  1  B1w       NA        0 125.09790 0.6362754 Inf 123.85083 126.3450
3  2  B3m 100.0000        0        NA        NA Inf        NA       NA
4  2  B1w       NA        0  94.47017 7.2279385 Inf  80.30367 108.6367
```

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(eUN.lmm)[1]
mu2 <- sum(coef(eUN.lmm)[1:2])
Omega_11 <- sigma(eUN.lmm)["B3m","B3m"]
Omega_21 <- sigma(eUN.lmm)["B1w","B3m"]
as.double(mu2 + Omega_21 * (100 - mu1) / Omega_11)
```

```
[1] 94.47017
```

5 Equivalence with other statistical methods

5.1 T-test

A t-test:

```
t.test(weight4 ~ group, data = gastricbypassW)
```

```


        Welch Two Sample t-test

data:  weight4 by group
t = 0.59144, df = 17.679, p-value = 0.5617
alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
95 percent confidence interval:
 -11.73582  20.91582
sample estimates:
mean in group 0 mean in group 1 
         104.66          100.07 
```

is equivalent to an independent covariance pattern with a different variable for each group:

```
e.ttest4 <- lmm(weight4 ~ group, structure = IND(~group),
               data = gastricbypassW, trace = FALSE)
model.tables(e.ttest4)
```

```
            estimate       se       df     lower     upper      p.value
(Intercept)   104.66 5.104469  9.00180  93.11324 116.20676 7.270954e-09
group          -4.59 7.760674 17.68244 -20.91558  11.73558 5.617090e-01
```

Multiple t-tests:

```
e.ttest1 <- lmm(weight1 ~ group, structure = IND(~group),
                data = gastricbypassW, trace = FALSE)
e.ttest2 <- lmm(weight2 ~ group, structure = IND(~group),
                data = gastricbypassW, trace = FALSE)
e.ttest3 <- lmm(weight3 ~ group, structure = IND(~group),
                data = gastricbypassW, trace = FALSE)
```

can be adjusted for multiple comparison by first using the `anova` function to specify the parameter of interest and combining the results using `rbind`:

```
e.mttest <- rbind(anova(e.ttest1, effects = "group=0"),
                  anova(e.ttest2, effects = "group=0"),
                  anova(e.ttest3, effects = "group=0"),
                  anova(e.ttest4, effects = "group=0"))
model.tables(e.mttest, method = "bonferroni")
```

```
               estimate       se       df     lower    upper p.value
weight1: group   -10.60 8.971747 17.96464 -35.49775 14.29775       1
weight2: group    -9.50 8.395143 17.98540 -32.79464 13.79464       1
weight3: group    -8.92 8.129458 17.95876 -31.48110 13.64110       1
weight4: group    -4.59 7.760674 17.68244 -26.16472 16.98472       1
```

⚠ efficient adjustment for multiple comparisons (like `"single-step"`) will not be valid as the correlation structure has not be specified. To do so it is more conveniently to work with a the long format:

```
e.mttest2 <- mlmm(weight ~ group, structure = IND(~group),
                  data = gastricbypassL, trace = FALSE,
                  effects = "group=0", by = "time", repetition = ~time|id)
model.tables(e.mttest2, method = "single-step2")
```

```
   by parameter estimate       se       df     lower     upper   p.value
1 B3m     group   -10.60 8.971747 17.96464 -30.90465  9.704648 0.3166268
2 B1w     group    -9.50 8.395143 17.98540 -28.49969  9.499691 0.3400966
3 A1w     group    -8.92 8.129458 17.95876 -27.31840  9.478400 0.3566864
4 A3m     group    -4.59 7.760674 17.68244 -22.15378 12.973775 0.6673833
```

or call the dedicated function `mt.test`:

```
mt.test(weight1+weight2+weight3+weight4~group, data = gastricbypassW)
```

```
       by parameter estimate       se       df     lower     upper   p.value
1 weight1     group   -10.60 8.971747 17.96464 -30.92628  9.726280 0.3200568
2 weight2     group    -9.50 8.395143 17.98540 -28.51993  9.519932 0.3433166
3 weight3     group    -8.92 8.129458 17.95876 -27.33800  9.498000 0.3602064
4 weight4     group    -4.59 7.760674 17.68244 -22.17249 12.992487 0.6654633
```

5.2 Linear regression on the change

A widely spread approach to analyze longitudinal data is to reduce the number of repetitions to 1 by working on the change and then apply 'usual' statistical methods. For instance one could compare the pre- and post- operation values using:

```
gastricbypassW$changeG41 <- gastricbypassW$glucagonAUC4-gastricbypassW$glucagonAUC1
e.change41 <- lm(changeG41 ~ weight1, data = gastricbypassW)
summary(e.change41)$coef
```

```
             Estimate Std. Error   t value   Pr(>|t|)
(Intercept) 17865.953 9292.61106  1.922598 0.07050076
weight1      -113.696   71.22173 -1.596367 0.12781371
```

This turns out to be equivalent to the following mixed model:

```
gastricbypassL41 <- gastricbypassL[gastricbypassL$visit %in% c(1,4),]
gastricbypassL41$time <- droplevels(gastricbypassL41$time)
gastricbypassL41$weight1 <- gastricbypassW$weight1[gastricbypassL41$id]

e.lmm41 <- lmm(glucagonAUC ~ time + time*weight1,
               repetition =~ time|id, structure = "UN",
               data = gastricbypassL41)
model.tables(e.lmm41)
```

```
                    estimate         se       df       lower       upper    p.value
(Intercept)      7730.051990 5737.22268 18.00298 -4323.26268 19783.36666 0.19458155
timeA3m         17865.953183 9292.61106 18.00104 -1657.01749 37388.92385 0.07049983
weight1             1.011014   43.97202 18.00298   -91.36968    93.39171 0.98190941
timeA3m:weight1  -113.695981   71.22173 18.00104  -263.32666    35.93469 0.12781271
```

This equivalence only holds as there is no missing data.

```
index.missing41 <- which(is.na(gastricbypassW$changeG41))
index.missing41
```

```
integer(0)
```

5.3 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 = 1.8667, df = 18, p-value = 0.07831
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.0484149  0.7174011
sample estimates:
      cor 
0.4027343
```

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) 13321.9427  5592.8058 2.381978 0.02845909
changeW41     380.3556   203.7541 1.866738 0.07831464
```

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  127.20
1.2  1      weight4  108.10
1.3  1 glucagonAUC1 5032.50
1.4  1 glucagonAUC4 9249.45
2.1  2      weight1  165.20
2.2  2      weight4  132.00
```

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    326.8357      415.3727    -46296.33
weight4         326.8357    290.8350    -5983.5871    -34434.03
glucagonAUC1    415.3727  -5983.5871 14299430.9269  -4229230.69
glucagonAUC4 -46296.3339 -34434.0320 -4229230.6877  20065722.32
```

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         48.01103      18261.26
d.glucagonAUC 18261.26175   42823614.62
```

and the corrrelation or covariance:

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

```
[1] 0.4027343
[1] 380.3556
```

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    0.4027343   0.1922078 2.660595    -0.2555602    1.061029 0.1386265
beta 380.3555798 198.3453360 2.798661 -277.3518013 1038.062961 0.1575655
```

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

6 Missing values and imputation

We reconsider the example of the previous section, but now in presence of missing values. The `summarize` function can be used to describe the amount of missing data at each repetition:

```
sss <- summarize(glucagon ~ time, data = gastricbypassL, na.rm = TRUE)
cbind(sss[,1:4], pc = paste0(100 * sss$missing / (sss$missing + sss$observed), "%"))
```

```
   outcome time observed missing pc
1 glucagon  B3m       20       0 0%
2 glucagon  B1w       19       1 5%
3 glucagon  A1w       19       1 5%
4 glucagon  A3m       20       0 0%
```

For more detail about the missing data patters, see the `summarizeNA` function:

```
summarizeNA(data = gastricbypassL, repetition = ~ time|id)
```

```
    variable frequency missing.pattern n.missing id B3m B1w A1w A3m
       visit        20           00000         0  0   0   0   0   0
      weight        20           00000         0  0   0   0   0   0
 glucagonAUC        18           00000         0  0   0   0   0   0
                     1           00100         1  0   0   1   0   0
                     1           00010         1  0   0   0   1   0
    baseline        20           00000         0  0   0   0   0   0
    glucagon        18           00000         0  0   0   0   0   0
                     1           00100         1  0   0   1   0   0
                     1           00010         1  0   0   0   1   0
       group        20           00000         0  0   0   0   0   0
```

To begin with we will only consider 1 week before and 1 week after surgery:

```
## long format
gastricbypassL32 <- gastricbypassL[gastricbypassL$visit %in% c(3,2),]
gastricbypassL32$time <- droplevels(gastricbypassL32$time)
gastricbypassL32$weight1 <- gastricbypassW$weight1[gastricbypassL32$id]
## wide format
gastricbypassW$changeG32 <- gastricbypassW$glucagonAUC3-gastricbypassW$glucagonAUC2
```

6.1 Full information approach

LMM uses a full information approach:

```
e.lmm32 <- lmm(glucagonAUC ~ time + time*weight1,
               repetition =~ time|id, structure = "UN",
               data = gastricbypassL32)
model.tables(e.lmm32)
```

```
                   estimate         se       df       lower       upper      p.value
(Intercept)      2226.30678 4973.21491 17.01148 -8265.72037 12718.33393 0.6600471546
timeA1w         37469.89400 8950.26818 17.88948 18657.74792 56282.04008 0.0005612515
weight1            37.90933   37.87004 17.01113   -41.98548   117.80414 0.3308362562
timeA1w:weight1  -213.20181   68.15807 17.71309  -356.56304   -69.84058 0.0058968630
```

whereas a linear model would perform a complete case approach:

```
e.change32 <- lm(changeG32 ~ weight1, data = gastricbypassW)
summary(e.change32)$coef
```

```
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 38101.9400 9417.61506  4.045816 0.0009373023
weight1      -217.2672   71.25218 -3.049271 0.0076504030
```

In the former the likelihood is evaluated using all observations, even those from individuals with some (but not all) missing outcome values: baseline is used even if follow-up is missing. In the later the likelihood is only evaluated on individuals with no missing outcome values: if follow-up is missing then baseline is not used. Indeed:

```
coef(lm(changeG32 ~ weight1, data = gastricbypassW[-c(5,15),]))
```

```
(Intercept)     weight1
 38101.9400   -217.2672
```

The estimates of the LMM can be retrived using a linear model where we have imputed the conditional expectation of the missing values given the observed value and the estimated model parameters: (see section 6.3 for a graphical representation)

```
gastricbypassWA <- fitted(e.lmm32, impute = TRUE, format = "wide")
gastricbypassWA$change32 <- gastricbypassWA$glucagonAUC_A1w - gastricbypassWA$glucagonAUC_B1w
gastricbypassWA$weight1 <- gastricbypassW$weight1[match(gastricbypassW$id,gastricbypassWA$id)]
coef(lm(change32 ~ weight1, data = gastricbypassWA))
```

```
(Intercept)     weight1
 37469.8940   -213.2018
```

⚠ Standard errors, confidence intervals, and p-values from this linear model should not be trusted as they do not account for the uncertainty in the imputed values.

6.2 Complete case approach

The `lmmCC` can be used to obtain the LMM that is equivalent to a linear regression. In the case of the comparing the change between groups, the `repetition` argument should indicate how the change has been computed:

```
e.lmmCC <- lmmCC(e.change32, repetition = changeG32 ~ glucagonAUC3-glucagonAUC2|id)
model.tables(e.lmmCC)
```

```
Remove 2 clusters (4 observations)
 - 2 observations with missing data (2 clusters)
 - 0 missing repetitions (0 clusters)
               estimate          se       df        lower       upper      p.value
(Intercept)  -36283.0356 12841.66795 15.99997 -63506.15929 -9059.91191 0.0121849728
time          38101.9400  9417.61506 16.00030  18137.51789 58066.36212 0.0009372705
weight1         257.7767    97.15802 15.99997     51.81085   463.74249 0.0173566171
time:weight1   -217.2672    71.25218 16.00030   -368.31484   -66.21956 0.0076502759
```

As output, the data from two clusters (i.e. 4 observations) has been excluded before fitting the LMM (instead of just the 2 observations with missing values for the full information approach). The interaction term of the LMM matches the regression coefficient of the linear model:

```
summary(e.change32)$coef
```

```
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 38101.9400 9417.61506  4.045816 0.0009373023
weight1      -217.2672   71.25218 -3.049271 0.0076504030
```

In the case of regressing two changes:

```
gastricbypassW$changeW32 <- gastricbypassW$weight3 - gastricbypassW$weight2

e2g.change32 <- lm(changeG32 ~ changeW32 + group, data = gastricbypassW)
summary(e2g.change32)$coef
```

```
             Estimate Std. Error    t value  Pr(>|t|)
(Intercept) 2720.5540   6930.588  0.3925430 0.7001787
changeW32   -783.8895   1122.541 -0.6983171 0.4956633
group       6059.7378   3525.140  1.7190062 0.1061756
```

the `repetition` argument should indicate how each change has been computed:

```
e2.lmmCC <-  lmmCC(e2g.change32, repetition = list(changeG32 ~ glucagonAUC3-glucagonAUC2|id,
                                                   changeW32 ~ weight3-weight2|id))
model.tables(e2.lmmCC)
```

```
Remove 2 clusters (8 observations)
 - 2 observations with missing data (2 clusters)
 - 0 missing repetitions (0 clusters)
         estimate           se       df        lower       upper   p.value
cor    -0.1774435    0.2416113 1.714523    -1.401851    1.046964 0.5499188
beta -783.8895353 1081.5567036 2.338122 -4848.482516 3280.703446 0.5342415
```

We retrieve the same estimate for the effect of change in weights but the uncertainty (standard error, confidence intervals, p.value) do not match. They should be asymptotically correct but may not have very good small smaple properties.

6.3 Imputation

When fitting a linear mixed model on a dataset with missing values:

```
eUN.lmmNA <- lmm(glucagon ~ time, repetition = ~time|id, data = gastricbypassL)
nobs(eUN.lmmNA)
```

```
obs         cluster     missing.obs missing.cluster
 78              20               2               0
```

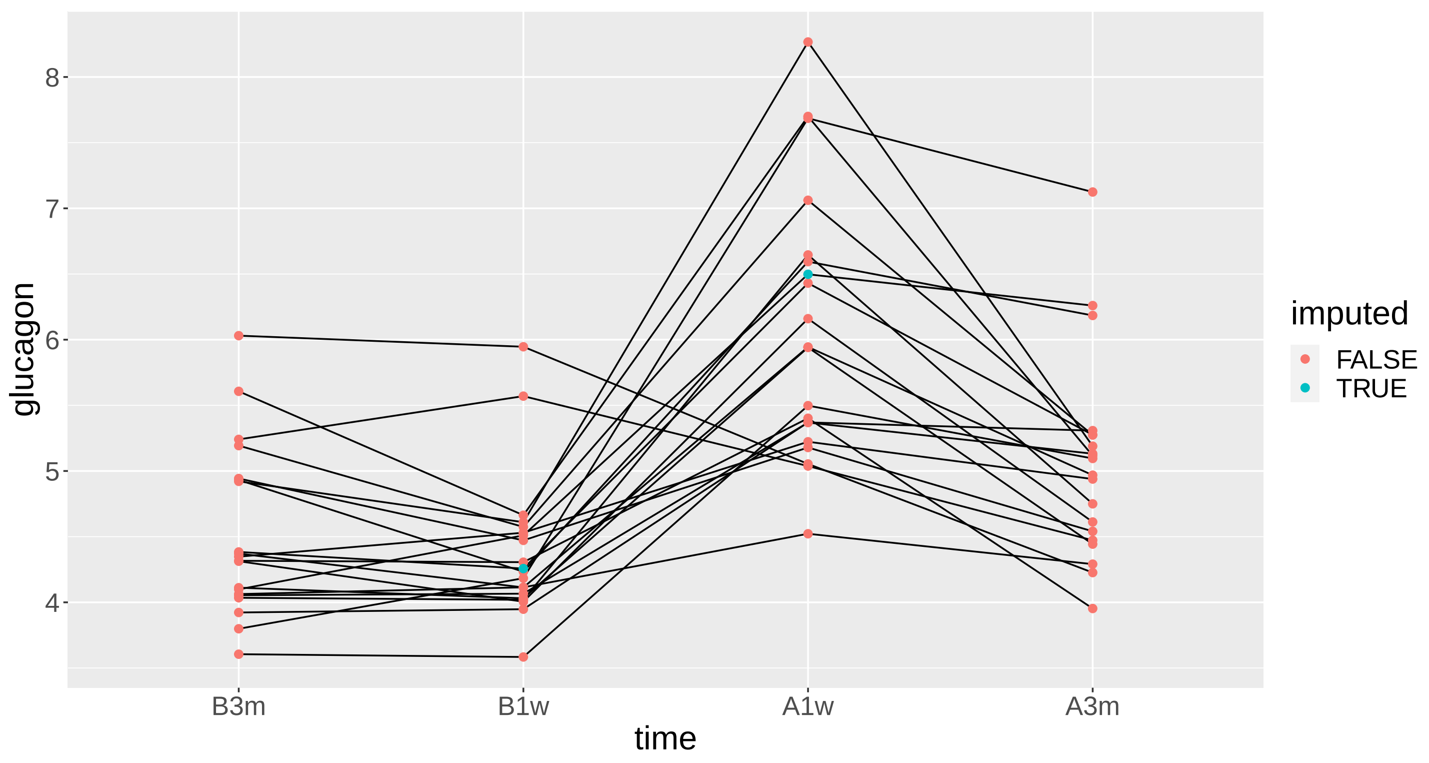
It is possible to extract the most likely value for these missing observations using the `fitted` function with argument `impute=TRUE`:

```
eData <- fitted(eUN.lmmNA, impute = TRUE, keep.newdata = TRUE)
eData$treat <- eData$treat2 <- eData$timeXtreat <- NULL
eData[eData$id %in% eData[eData$imputed,"id"],]
```

```
   id visit time weight glucagonAUC baseline glucagon group imputed
5   5     1  B3m  113.1      7090.5     TRUE 4.383738     1   FALSE
15 15     1  B3m  115.0      5410.5     TRUE 4.098741     1   FALSE
25  5     2  B1w  105.6          NA     TRUE 4.256984     1    TRUE
35 15     2  B1w  109.7      7833.0     TRUE 4.509697     1   FALSE
45  5     3  A1w   99.9     19155.0    FALSE 6.430376     1   FALSE
55 15     3  A1w  103.5          NA    FALSE 6.497856     1    TRUE
65  5     4  A3m   87.7     12345.0    FALSE 5.275118     1   FALSE
75 15     4  A3m   94.1     18148.5    FALSE 6.259632     1   FALSE
```

Missing outcome values in the dataset have been replaced by its most likely value (which is the same as the dynamic prediction, describedy previously). A column `imputed` has also been added to differentiate between the the modeled and observed value. Visually:

```
ggplot(eData, aes(x=time,y=glucagon, group=id)) + geom_line() + geom_point(aes(color=imputed))
```



It is possible to sample from the estimated distribution of the missing value instead of using the most likely value, e.g. accounting for residual variance and uncertainty related to parameter estimation:

```
set.seed(10)
index.na <- which(is.na(gastricbypassL$glucagonAUC))
fitted(eUN.lmmNA, impute = TRUE, se.impute = "total")[index.na]
fitted(eUN.lmmNA, impute = TRUE, se.impute = "total")[index.na]
fitted(eUN.lmmNA, impute = TRUE, se.impute = "total")[index.na]
```

```
[1] 4.262434 6.305287
[1] 3.858267 5.871642
[1] 4.342624 6.905246
```

6.4 Multiple imputation

The `mlmm` function can used to perform stratify analyses, typically useful when performing multiple imputations. Consider the wide format of the dataset where a few values are missing:

```
data(gastricbypassW, package = "LMMstar")
colSums(is.na(gastricbypassW))
```

```
          id      weight1      weight2      weight3      weight4 glucagonAUC1 glucagonAUC2
           0            0            0            0            0            0            1
glucagonAUC3 glucagonAUC4
           1            0
```

We use `mice` to generate a number of imputed datasets (here 5):

```
library(mice)
set.seed(10)
gastricbypassW.mice <- mice(gastricbypassW, m = 5, printFlag = FALSE)
gastricbypassW.NNA <- complete(gastricbypassW.mice, action = "long")
table(gastricbypassW.NNA$.imp)
```

```
Advarselsbesked:
Number of logged events: 110

 1  2  3  4  5
20 20 20 20 20
```

We can then use `mlmm` to perform a separate linear regression per dataset:

```
e.mlmm <- mlmm(glucagonAUC3~glucagonAUC2+weight2, data=gastricbypassW.NNA,
               by = ".imp", effects = "weight2=0", trace = FALSE)
model.tables(e.mlmm)
```

```
  by parameter  estimate       se      df     lower     upper     p.value
1  1   weight2 -204.6291 62.88617 17.0034 -337.3053 -71.95289 0.004670840
2  2   weight2 -194.4004 62.31006 17.0034 -325.8611 -62.93968 0.006231893
3  3   weight2 -211.9042 65.51654 17.0034 -350.1299 -73.67848 0.004872354
4  4   weight2 -199.8417 62.12071 17.0034 -330.9029 -68.78041 0.005058119
5  5   weight2 -199.9269 62.16057 17.0034 -331.0722 -68.78152 0.005065662
```

and pool the results using Rubin's rule:

```
model.tables(e.mlmm, method = "pool.rubin")
```

```
        estimate      se       df     lower     upper     p.value
<1, 5> -202.1404 63.4192 15.09811 -337.2388 -67.04208 0.006078676
```

This matches[4] the results obtained with the mice package:

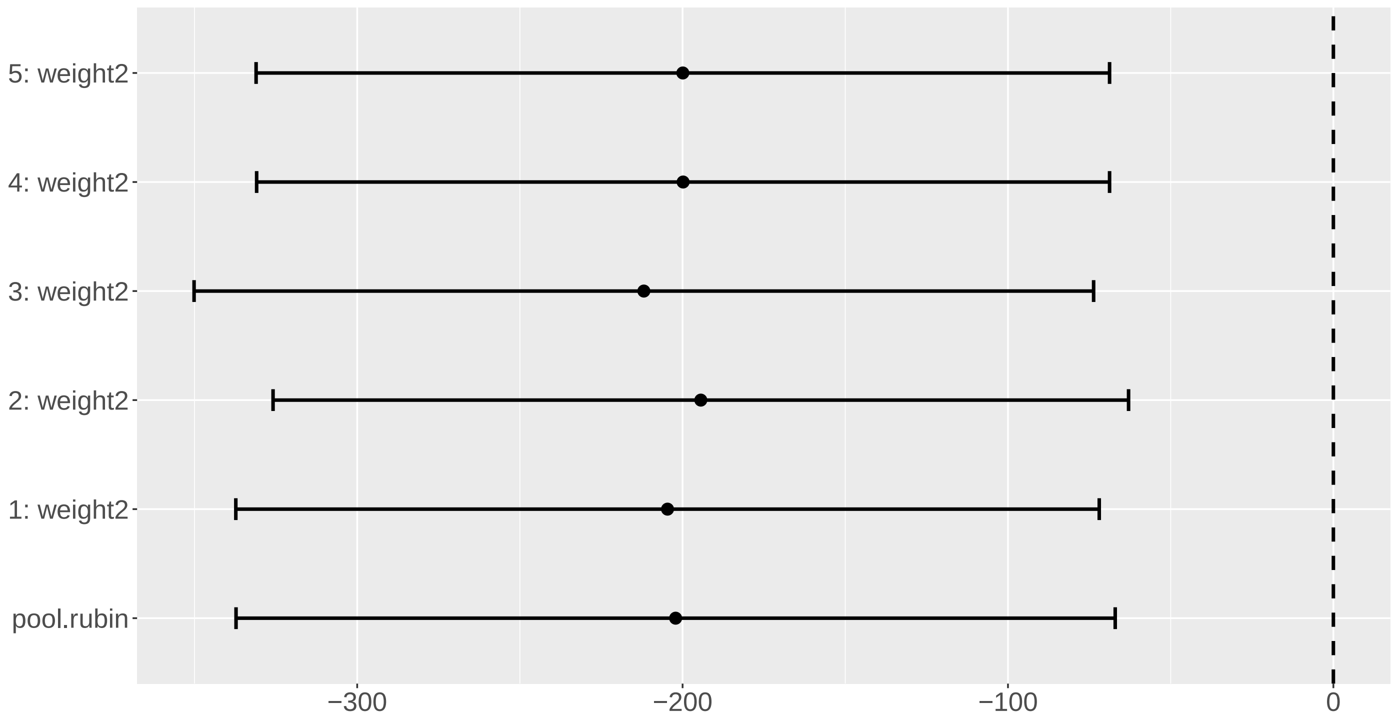
[4]almost exactly, only the degrees of freedom are a little different

```
e.mice <- with(data=gastricbypassW.mice,exp=lm(glucagonAUC3~glucagonAUC2+weight2))
summary(pool(e.mice))
```

```
         term      estimate    std.error  statistic       df      p.value
1  (Intercept)  4.119699e+04 7674.2675772  5.3681988 15.08457 7.675819e-05
2 glucagonAUC2  7.038742e-02    0.3689445  0.1907805 15.23549 8.512165e-01
3      weight2 -2.021404e+02   63.4191998 -3.1873698 15.09481 6.080058e-03
```

One can use the `plot` function to obtain a forest plot of the individual estimates along with the pooled estimate:

```
plot(e.mlmm, method = c("pool.rubin","none"))
```



7 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
```

8 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)
```

9 R session

Details of the R session used to generate this document:

```
sessionInfo()
```

```
R version 4.2.0 (2022-04-22 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 19045)

Matrix products: default

locale:
[1] LC_COLLATE=Danish_Denmark.utf8  LC_CTYPE=Danish_Denmark.utf8    LC_MONETARY=Danish_Denmark.utf8
[4] LC_NUMERIC=C                    LC_TIME=Danish_Denmark.utf8

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

other attached packages:
 [1] mice_3.14.0         sandwich_3.0-2      scales_1.2.1        rlang_1.1.1
 [5] pbapply_1.7-0       numDeriv_2016.8-1.1 nlme_3.1-158        lava_1.7.2.1
 [9] doSNOW_1.0.20       snow_0.4-4          iterators_1.0.14    foreach_1.5.2
[13] copula_1.1-2        lme4_1.1-29         Matrix_1.5-1        LMMstar_1.0.0
[17] ggpubr_0.4.0        multcomp_1.4-22     TH.data_1.1-1       MASS_7.3-57
[21] survival_3.3-1      mvtnorm_1.2-3       qqtest_1.2.0        emmeans_1.8.8-090002
[25] ggplot2_3.4.3

loaded via a namespace (and not attached):
 [1] butils.base_1.2     minqa_1.2.4         colorspace_2.1-0    ggsignif_0.6.3
 [5] ellipsis_0.3.2      estimability_1.4.1  parameters_0.18.2   fs_1.6.3
 [9] listenv_0.9.0       farver_2.1.1        remotes_2.4.2       gsl_2.1-8
[13] fansi_1.0.4         codetools_0.2-18    splines_4.2.0       doParallel_1.0.17
[17] cachem_1.0.8        pkgload_1.3.0       nloptr_2.0.3        broom_0.8.0
[21] stabledist_0.7-1    effectsize_0.7.0.5  shiny_1.7.2         compiler_4.2.0
[25] backports_1.4.1     fastmap_1.1.1       cli_3.6.1           later_1.3.0
[29] htmltools_0.5.6     prettyunits_1.1.1   tools_4.2.0         lmerTest_3.1-3
[33] coda_0.19-4         gtable_0.3.4        glue_1.6.2          reshape2_1.4.4
[37] dplyr_1.1.3         Rcpp_1.0.11         carData_3.0-5       vctrs_0.6.3
[41] insight_0.18.4      stringr_1.5.0       globals_0.16.2      ps_1.7.1
[45] mime_0.12           miniUI_0.1.1.1      lifecycle_1.0.3     devtools_2.4.4
[49] rstatix_0.7.0       future_1.31.0       zoo_1.8-11          promises_1.2.0.1
[53] memoise_2.0.1       gridExtra_2.3       stringi_1.7.12      bayestestR_0.13.0
[57] pcaPP_2.0-3         boot_1.3-28         pkgbuild_1.3.1      pkgconfig_2.0.3
[61] lattice_0.20-45     purrr_1.0.2         htmlwidgets_1.6.2   labeling_0.4.3
[65] cowplot_1.1.1       tidyselect_1.2.0    processx_3.6.1      parallelly_1.34.0
[69] plyr_1.8.7          magrittr_2.0.3      R6_2.5.1            generics_0.1.3
[73] profvis_0.3.7       ADGofTest_0.3       pillar_1.9.0        withr_2.5.1
```

```
[77] mgcv_1.8-40         datawizard_0.6.1    abind_1.4-5         pspline_1.0-19
[81] tibble_3.2.1        future.apply_1.10.0 crayon_1.5.1        car_3.1-0
[85] utf8_1.2.3          urlchecker_1.0.1    usethis_2.1.6       data.table_1.14.2
[89] callr_3.7.2         digest_0.6.33       xtable_1.8-4        tidyr_1.3.0
[93] httpuv_1.6.5        stats4_4.2.0        munsell_0.5.0       sessioninfo_1.2.2
```

References

Christensen, R. (2011). *Plane answers to complex questions (4th edition)*, volume 35. Springer.

Geyer, C. J. (2013). Asymptotics of maximum likelihood without the lln or clt or sample size going to infinity. In *Advances in Modern Statistical Theory and Applications: A Festschrift in honor of Morris L. Eaton*, volume 10, pages 1–25. Institute of Mathematical Statistics.

Lakens, D. (2013). Calculating and reporting effect sizes to facilitate cumulative science: a practical primer for t-tests and anovas. *Frontiers in psychology*, 4:863.

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

Pipper, C. B., Ritz, C., and Bisgaard, H. (2012). A versatile method for confirmatory evaluation of the effects of a covariate in multiple models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 61(2):315–326.

Appendix A Likelihood in a linear mixed model

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. We consider n replicates (i.e. $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$) and $VX_1, \ldots, \boldsymbol{X}_n$) along with a vector of weights $\omega = (w_1, \ldots, w_n)$, which are by default all equal to 1.

A.1 Log-likelihood

The restricted log-likelihood in a linear mixed model can then be written:

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

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[5]. 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 $\theta \in \boldsymbol{\Theta}$:

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

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`.

[5]The REML is the likelihood of the observations divided by the prior on the estimated mean parameters $\widehat{\boldsymbol{\Theta}}_\mu \sim \mathcal{N}(\mu, \left(\boldsymbol{X}\Omega^{-1}(\boldsymbol{\Theta})\boldsymbol{X}^\intercal\right)^{-1})$. 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, $\mathcal{H}(\boldsymbol{\Theta})$, with element[6]:

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

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

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:

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

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

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

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

[6]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:

$$df\left(\sigma_{\widehat{\beta}}\right) = \frac{2\sigma^4_{\widehat{\beta}}}{\mathbb{V}ar\left[\widehat{\sigma}_{\widehat{\beta}}\right]}$$

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 obtain the later we first notice that:

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

where

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

So we will first derive the derivative of $\mathbb{E}\left[\mathcal{H}(\theta,\theta')|\boldsymbol{X}\right]$ and then the one of the blue term in Equation C. 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 first notice that the derivative with respect to the mean parameters is 0. So we just need to compute the derivative with respect to a variance parameter θ'':

$$\begin{aligned}
&\frac{\partial A(\boldsymbol{\Theta})^{-1}\left(\sum_{i=1}^n w_i b_i(\boldsymbol{\Theta}) B_i(\boldsymbol{\Theta}) b_i^\intercal(\boldsymbol{\Theta}) + C(\boldsymbol{\Theta})A(\boldsymbol{\Theta})^{-1}C^\intercal(\boldsymbol{\Theta})\right)}{\partial\theta''}\\
=&A(\boldsymbol{\Theta})^{-1}\frac{\partial A(\boldsymbol{\Theta})}{\partial\theta''}A(\boldsymbol{\Theta})^{-1}\left(\sum_{i=1}^n w_i b_i(\boldsymbol{\Theta}) B_i(\boldsymbol{\Theta}) b_i^\intercal(\boldsymbol{\Theta}) + C(\boldsymbol{\Theta})A(\boldsymbol{\Theta})^{-1}C^\intercal(\boldsymbol{\Theta})\right)\\
&+A(\boldsymbol{\Theta})^{-1}\left(\sum_{i=1}^n w_i\left(\frac{\partial b_i(\boldsymbol{\Theta})}{\partial\theta''}B_i(\boldsymbol{\Theta})b_i^\intercal(\boldsymbol{\Theta}) + b_i(\boldsymbol{\Theta})\frac{\partial B_i(\boldsymbol{\Theta})}{\partial\theta''}b_i^\intercal(\boldsymbol{\Theta}) + b_i(\boldsymbol{\Theta})B_i(\boldsymbol{\Theta})\frac{\partial b_i^\intercal(\boldsymbol{\Theta})}{\partial\theta''}\right.\right.\\
&\qquad\left.\left.+\frac{\partial C(\boldsymbol{\Theta})}{\partial\theta''}A^{-1}(\boldsymbol{\Theta})C^\intercal(\boldsymbol{\Theta}) + C(\boldsymbol{\Theta})A^{-1}\frac{\partial A(\boldsymbol{\Theta})}{\partial\theta''}A^{-1}C^\intercal(\boldsymbol{\Theta}) + C(\boldsymbol{\Theta})A^{-1}(\boldsymbol{\Theta})\frac{\partial C^\intercal(\boldsymbol{\Theta})}{\partial\theta''}\right)\right)
\end{aligned}$$

and

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

where:

$$\begin{aligned}
\frac{\partial A(\boldsymbol{\Theta})}{\partial\theta''} &= \sum_{i=1}^n w_i\boldsymbol{X}_i\Omega_i^{-1}(\boldsymbol{\Theta})\frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta''}\Omega_i^{-1}(\boldsymbol{\Theta})\boldsymbol{X}_i^\intercal\\
\frac{\partial b_i(\boldsymbol{\Theta})}{\partial\theta''} &= \boldsymbol{X}_i\Omega_i^{-1}(\boldsymbol{\Theta})\frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta''}\Omega_i^{-1}(\boldsymbol{\Theta})\\
\frac{\partial B_i(\boldsymbol{\Theta})}{\partial\theta''} &= \frac{\partial^3\Omega_i(\boldsymbol{\Theta})}{\theta\theta'\theta''}\\
&\quad-2\left(\frac{\partial^2\Omega_i(\boldsymbol{\Theta})}{\partial\theta\partial\theta''}\Omega_i^{-1}(\boldsymbol{\Theta})\frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta'} + \frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta}\Omega_i^{-1}(\boldsymbol{\Theta})\frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta''}\Omega_i^{-1}(\boldsymbol{\Theta})\frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta'} + \frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta}\Omega_i^{-1}(\boldsymbol{\Theta})\frac{\partial^2\Omega_i(\boldsymbol{\Theta})}{\partial\theta'\partial\theta''}\right)\\
\frac{\partial C(\boldsymbol{\Theta})}{\partial\theta''} &= \sum_{i=1}^n w_i\boldsymbol{X}_i\Omega_i^{-1}(\boldsymbol{\Theta})\left(\frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta''}\Omega_i^{-1}(\boldsymbol{\Theta})\frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta} + \frac{\partial^2\Omega_i(\boldsymbol{\Theta})}{\partial\theta\partial\theta''} + \frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta}\Omega_i^{-1}(\boldsymbol{\Theta})\frac{\partial\Omega_i(\boldsymbol{\Theta})}{\partial\theta''}\right)\Omega_i^{-1}(\boldsymbol{\Theta})\boldsymbol{X}_i^\intercal
\end{aligned}$$

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.
Let's take an example:

```
## data(dfL, package = "LMMstar")
dfTest <- dfL
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:

```
eML.lmmUN <- lmm(weight ~ time+glucagon, data = dfTest, repetition = ~time|id, method = "ML")
eML.lmmUN2 <- lmm(weight ~ time+glucagon2, data = dfTest, repetition = ~time|id, method = "ML
   ")
```

```
logLik(eML.lmmUN)
logLik(eML.lmmUN2)
```

```
[1] -218.71
[1] -218.71
```

but it does when using REML:

```
eREML.lmmUN <- lmm(weight ~ time + glucagon, data = dfTest, repetition = ~time|id, method = "
   REML")
eREML.lmmUN2 <- lmm(weight ~ time + glucagon2, data = dfTest, repetition = ~time|id, method =
    "REML")
```

```
logLik(eREML.lmmUN)-logLik(eREML.lmmUN2)
log(2)
```

```
[1] 0.6931472
[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(5)
dfTest$ff <- rbinom(NROW(dfTest), size = 1, prob = 0.5)
logLik(lmm(weight ~ time+glucagon, data = dfTest, repetition = ~time|id, method = "REML"))
logLik(lmm(weight ~ time+glucagon*ff, data = dfTest, repetition = ~time|id, method = "REML"))
```

```
[1] -216.3189
[1] -216.8425
```

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

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

```
[1] -218.71
[1] -218.6259
```

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.

Appendix C Sum of squares in a linear mixed model

All mixed models implemented in LMMstar can be written as:

$$Y_{it} = X_{it}\beta + \varepsilon_{it} \text{ where } \varepsilon_i \sim \mathcal{N}(0, \Omega)$$

where Y denote the outcome repeteadly measured within each cluster i where t indexes the repetitions. X denotes the covariates, β the mean parameters, ε the residuals, and Ω the residual variance-covariance matrix. Ω must be positive definite so there must exist a square postive definite matrix $\Omega^{1/2}$ such that $\Omega^{1/2}\Omega^{1/2} = \Omega$. Therefore the previous model is equivalent to:

$$Y^*_{it} = X^*_{it}\beta + \varepsilon^*_{it} \text{ where } \varepsilon_i \sim \mathcal{N}(0, I_T)$$

where $Y^*_i = \Omega^{-1/2}Y_i$, $X^*_i = \Omega^{-1/2}X_i$, $\varepsilon^*_i = \Omega^{-1/2}\varepsilon_i$, and I_x is the identity matrix with x rows and columns. One can then introduce the projectors $H = X\left(X^\intercal\Omega^{-1}X\right)^{-1}X^\intercal\Omega^{-1}$ and $H^* = X^*\left(X^{*\intercal}X^*\right)^{-1}X^{*\intercal}$ onto the space spanned by X and X^* respectively. We can now define the "normalized" residual sum of squares as the squared sum of the normalized residuals:

$$\begin{aligned} SSE^* = \varepsilon^{*\intercal}\varepsilon^* &= Y^{*\intercal}(I_{nT} - H^*)Y^* \\ &= Y^\intercal\Omega^{-1}Y - Y^\intercal\Omega^{-1}X\left(X^\intercal\Omega^{-1}X\right)^{-1}X^\intercal\Omega^{-1}Y \\ &= Y^\intercal(I_{nT} - H^\intercal)\Omega^{-1}(I_{nT} - H)Y \end{aligned}$$

The previous to last line uses that: $(I_{nT} - H^\intercal)\Omega^{-1}(I_{nT} - H) = \Omega^{-1} - H^\intercal\Omega^{-1} - \Omega^{-1}H + H^\intercal\Omega^{-1}H = \Omega^{-1} - H^\intercal\Omega^{-1}$ as $H^\intercal\Omega^{-1}H = \Omega^{-1}HH = \Omega^{-1}H$ since H is a projector. Note that compared to the "traditional" SSE defined for linear regression and random effect models (e.g. see Christensen (2011) section 2.7), $SSE = \delta SSE^*$ where δ is the residual variance conditional on any random effects, i.e. SSE^* are the residual degrees of freedom. This is because the same definition for the sum of squares is used except that $\varepsilon_i \sim \mathcal{N}(0, \delta\Omega)$.

We can also define the "normalized" regression sum of squares:

$$\begin{aligned} SSR^* = (X^*\beta)^\intercal X^*\beta = (H^*Y^*)^\intercal H^*Y^* &= Y^{*\intercal}H^*Y^* \\ &= Y^\intercal H^\intercal\Omega^{-1}Y^* = Y^\intercal H^\intercal H^\intercal\Omega^{-1}Y^* = Y^\intercal H^\intercal\Omega^{-1}HY^* \\ &= \widehat{\beta}X^\intercal\Omega^{-1}X\widehat{\beta} \end{aligned}$$

where $\widehat{\beta} = \left(X^\intercal\Omega^{-1}X\right)^{-1}X^\intercal\Omega^{-1}Y$. Note that when using the expected information $SSR^* = \widehat{\beta}\Sigma^{-1}_{\widehat{\beta}}\widehat{\beta}$, i.e. it is the F-statistics times the number of parameters. Again the "traditional" SSR defined for linear regression and random effect models is proportional to this normalized SSR: $SSR = \delta SSR^*$.

The proportion of explained variance of p parameters can thus be re-expressed as:

$$R^2 = \frac{SSR}{SSR + SSE} = \frac{SSR^*}{SSR^* + SSE^*} = \frac{Fp}{Fp + df}$$

where df denotes the residual degrees of freedom, typically $n - p$ in a univariate linear model fitted with n observations.

⚠ In practice *df* is estimated using the Satterthwaite approximation of the degrees of freedom of the regression coefficient. This is only equivalent to the "SSR/SSE" formula in univariate linear regression.

Illustration for a univariate linear model:

Data without missing values:

```
df.aov <- dfL[!is.na(dfL$glucagon),]
```

Traditional anova decomposition:

```
e.lm <- lm(weight ~ time + glucagon, data = df.aov)
car::Anova(e.lm, type = "II")
```

```
Anova Table (Type II tests)

Response: weight
           Sum Sq Df F value    Pr(>F)
time       6367.3  3  6.4308 0.0006329 ***
glucagon   1964.8  1  5.9531 0.0171207 *
Residuals 24093.1 73
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Fit `lmm`:

```
e.lmm <- lmm(weight ~ time + glucagon, data = df.aov)
```

Residual sum of squares (SSE):

```
SSEstar <- crossprod(residuals(e.lmm, type = "normalized"))
c(SSEstar = SSEstar, SSE = SSEstar * sigma(e.lmm))
```

```
SSEstar      SSE
  73.00 24093.11
```

The normalized SSE can also be obtained using the `df.residual` method:

```
df.residual(e.lmm)
```

```
[1] 73
```

Regression sum of squares (SSR):

```
eBeta.lmm <- coef(e.lmm)
eVcov.lmm <- vcov(e.lmm, type.information = "expected")

SSRstar.glucagon <- eBeta.lmm[5] %*% solve(eVcov.lmm[5,5]) %*% eBeta.lmm[5]
SSRstar.time <- eBeta.lmm[2:4] %*% solve(eVcov.lmm[2:4,2:4]) %*% eBeta.lmm[2:4]
c(SSR.glucagon = SSRstar.glucagon * sigma(e.lmm),
  SSR.time = SSRstar.time * sigma(e.lmm),
  F.glucagon = SSRstar.glucagon,
  F.time = SSRstar.time/3)
```

```
SSR.glucagon      SSR.time   F.glucagon       F.time
 1964.764452  6367.324429     5.953062     6.430810
```

So the proportion of explained variance is:

```
R2.glucagon <- SSRstar.glucagon/(SSRstar.glucagon+SSEstar)
R2.glucagon
```

```
           [,1]
[1,] 0.07540002
```

and the corresponding partial correlation is:

```
sign(coef(e.lmm)["glucagon"])*sqrt(R2.glucagon)
```

```
           [,1]
[1,] -0.2745906
```

which matches the output of `partialCor`:

```
summary(partialCor(e.lmm, R2 = TRUE))
```

```
                Partial correlation

          estimate    se df  lower  upper  p.value
   timeB1w    -0.153 0.113 73 -0.378  0.072   0.1796
   timeA1w    -0.038 0.117 73  -0.27  0.195   0.7475
   timeA3m    -0.413 0.088 73 -0.589 -0.236 1.36e-05
   glucagon   -0.275 0.104 73 -0.482 -0.067   0.0102
   ------------------------------------------------
  Columns lower and upper contain 95% pointwise confidence intervals for each coefficient.
  Degrees of freedom were computed using a Satterthwaite approximation (column df).

                Coefficient of determination (R2)

          estimate    se df  lower upper  p.value
   time        0.209 0.075 73  0.059 0.359 0.006976
   glucagon    0.075 0.057 73 -0.038 0.189 0.191156
   global      0.285 0.076 73  0.134 0.435 0.000328
   ------------------------------------------------
  Columns lower and upper contain 95% pointwise confidence intervals for each coefficient.
  Degrees of freedom were computed using a Satterthwaite approximation (column df).
```

Appendix D Equivalence with other R packages

D.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):

```
eCS.gls <- gls(weight ~ time + glucagon, correlation = corCompSymm(form=~time|id),
               data = dfL, na.action = na.omit)
eCS.lme <- lme(weight ~ time + glucagon, random = ~1|id,
               data = dfL, na.action = na.omit)
logLik(eCS.lme)
logLik(eCS.gls)
logLik(eCS.lmm)
```

```
'log Lik.' -243.6005 (df=7)
'log Lik.' -243.6005 (df=7)
[1] -243.6005
```

The estimated random effect also match:

```
range(ranef(eCS.lmm)$estimate-ranef(eCS.lme))
```

```
[1] -3.136991e-08  2.384372e-08
```

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

```
eUN.gls <- gls(weight ~ time + glucagon,
               correlation = corSymm(form=~as.numeric(time)|id),
               weights = varIdent(form=~1|time),
               data = dfL, na.action = na.omit)
logLik(eUN.gls)
logLik(eUN.lmm)
```

```
'log Lik.' -216.3189 (df=15)
[1] -216.3189
```

D.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 ~ time + glucagon + (1|id),
                 data = dfL)
logLik(eRI.lmer)
logLik(eRI.lmm)
```

```
'log Lik.' -243.6005 (df=7)
[1] -243.6005
```

The estimated random effects match:

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

```
[1] -3.167867e-08  2.406756e-08
```

Nested random effects correspond to block unstructured:

```
eNRI.lmer <- lmer(weight ~ time*group + (1|id/baseline),
                  data = dfL)
logLik(eNRI.lmer)
logLik(eNRI.lmm)
```

```
'log Lik.' -230.5328 (df=11)
[1] -230.5328
```

And the estimated random effects still match:

```
eRanefNRI.lmm <- ranef(eNRI.lmm)
eRanefNRI.lmer <- ranef(eNRI.lmer)
## id
range(eRanefNRI.lmm[eRanefNRI.lmm$variable=="id","estimate"]-eRanefNRI.lmer$id)
## baseline
range(eRanefNRI.lmm[eRanefNRI.lmm$variable!="id","estimate"]-ranef(eNRI.lmer)$`baseline:id`)
```

```
[1] -7.457484e-05  1.182242e-04
[1] -0.0001493705  0.0001080902
```

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

```
eUN.lmer <- lmer(weight ~ time + glucagon + (0 + time|id),
                 data = dfL, control = lmerControl(check.nobs.vs.nRE = "ignore"))
logLik(eUN.lmer)
logLik(eUN.lmm)
```

```
'log Lik.' -216.3189 (df=16)
[1] -216.3189
```

Note that however the uncertainty is quantified in a slightly different way, e.g.:

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

```
             Multivariate Wald test

               F-statistic       df  p.value
mean: time          86.743 (3,19.0) 2.84e-11 ***
    : glucagon      13.518 (1,13.7)  0.00257  **
```

do not match

```
anova(eUN.lmer)
```

```
Type III Analysis of Variance Table with Satterthwaite's method
          Sum Sq Mean Sq NumDF  DenDF F value    Pr(>F)
time     114.275  38.092     3 20.483  87.242 7.784e-12 ***
glucagon  10.125  10.125     1 16.784  23.191 0.0001671 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

I think this is because `lmer` base uncertainty computation on the expected information (instead of the observed information). Doing so leads to more similar results:

```
eUN2.lmm <- lmm(weight ~ time + glucagon, repetition = ~time|id,
                structure = "UN", data = dfL, type.information = "expected")
suppressWarnings(anova(eUN2.lmm))
```

```
             Multivariate Wald test

               F-statistic       df  p.value
mean: time          87.253 (3,22.5) 1.48e-12 ***
    : glucagon      23.198 (1,19.4) 0.000114 ***
```

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.' -165.4303 (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] -165.4303
```

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] -4.381305e-07  6.017161e-07
```

D.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
)
```

```
mmrm() registered as emmeans extension
```

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"
)
```

```
Advarselsbesked:
I .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] -2.541278e-06
[1] -0.0001830095  0.0001626755
[1] -0.0003971008  0.0002047941
```

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 memory 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.

D.4 effectsize package (R^2 or η^2)

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

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

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

Parameter | Eta2 (partial) |       95% CI
-----------------------------------------
time      |           0.92 | [0.89, 1.00]
glucagon  |           0.03 | [0.00, 1.00]

- One-sided CIs: upper bound fixed at [1.00].>
```

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] 0.92380363 0.03162017
```

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] 0.9319379 0.4965135
```

compared to:

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

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

Parameter | Eta2 (partial) |       95% CI
-----------------------------------------
time      |           0.93 | [0.87, 1.00]
glucagon  |           0.58 | [0.29, 1.00]

- One-sided CIs: upper bound fixed at [1.00].>
```

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

D.5 MuMIn package (R^2)

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

```
           R2m       R2c
[1,] 0.2163302 0.9764382
```

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
0.2163302 0.9764382
```

D.6 stats package (partial residuals)

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

```
eIID.lm <- lm(weight ~ time + glucagon, data = dfL)
pRes.lm <- residuals(eIID.lm, type = "partial")
head(pRes.lm)
```

```
       time   glucagon
1  3.359543   1.475108
2 49.419060  39.475108
3 -8.145299 -16.024892
4 24.241798  20.475108
5 -8.407618 -12.624892
6 41.027985  33.075108
```

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

```
eIID.lmm <- lmm(weight ~ time + glucagon, data = dfL)
head(residuals(eIID.lmm, type = "partial", var = "glucagon"))
```

```
[1] -31.9349871   6.0650129 -49.4349871 -12.9349871 -46.0349871  -0.3349871
```

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

```
m.pres2 <- dfL$weight - cbind(model.matrix(~time,dfL), mean(dfL$glucagon)) %*% coef(eIID.lmm)
range(pRes.lm[,"glucagon"] - m.pres2, na.rm = TRUE)
```

```
[1] -3.348433e-13  4.654055e-13
```

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

```
eIID.lmm <- lmm(weight ~ time + glucagon, data = dfL)
pRes.lmm <- residuals(eIID.lmm, type = "partial-center", var = "glucagon")
range(pRes.lm[,"glucagon"]-pRes.lmm)
```

```
[1] -3.330669e-13  4.725109e-13
```

⚠ 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.