Overview of the package LMMstar

Brice Ozenne

March 29, 2023

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

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

- the function `lmm` is the main function of the package which fits linear mixed models. The user can interact with *lmm* objects using:
  - `anova` to test linear combinations of coefficients (Wald test or Likelihood ratio tests). The output be combined via `rbind`.
  - `coef` to extract the estimates.
  - `confint` to extract the estimates with their confidence intervals.
  - `dummy.coef` to extract the estimated (marginal) mean for each combination of categorical covariate.
  - `estimate` to test non-linear combinations of coefficients (Wald test via a first order delta method).
  - `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 conditional mean for new observations.
  - `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.
  - `sigma` to extract the modeled residual variance covariance matrix.
  - `summary` to obtain a summary of the input, model fit, and estimated values.

- the `mlmm` function to fit (distinct) linear mixed models on different outcome, and gather the estimated coefficients.
- the `summarize` function to compute summary statistics, possibly stratified on a categorical variable
- the `summarizeNA` function to identify missing data patterns.
- the `partialCor` function to compute partial correlation between two variables.
- the `sampleRem` function to simulate longitudinal data.
- the `LMMstar.options` function enables the user to display the default values used in the **LMMstar** package. The function can also change the default values to better match the user needs.

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

```
library(LMMstar)
```

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

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

```
[1] '0.9.0'
```

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 3monthsBefore  127.2     5032.50
2  2     1 3monthsBefore  165.2    12142.50
3  3     1 3monthsBefore  109.7    10321.35
4  4     1 3monthsBefore  146.2     6693.00
5  5     1 3monthsBefore  113.1     7090.50
6  6     1 3monthsBefore  158.8    10386.00
```

See ?gastricbypassL for a presentation of the dataset. We will shorten the values of the time variable:

```
gastricbypassL$time <- factor(gastricbypassL$time,
                              levels = c("3monthsBefore", "1weekBefore",
                                         "1weekAfter", "3monthsAfter" ),
                              labels = c("B3m","B1w","A1w","A3m"))
gastricbypassL$visit <- as.numeric(gastricbypassL$time) ## convert to numeric
gastricbypassL$baseline <- gastricbypassL$visit<=2
```

rescale the glucagon values

```
gastricbypassL$glucagon <- as.double(scale(gastricbypassL$glucagonAUC))+5
```

and add a group variable:

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

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      5032.50       4942.5      20421.0      9249.45
2  2   165.2   153.4   149.2   132.0     12142.50      14083.5      10945.5      7612.50
3  3   109.7   101.6    97.7    87.1     10321.35       6202.5      20121.0     17704.50
4  4   146.2   142.4   136.7   123.0      6693.00       6631.5      13090.5      4551.00
5  5   113.1   105.6    99.9    87.7      7090.50           NA      19155.0     12345.00
6  6   158.8   143.6   134.6   108.7     10386.00       7609.5      11778.0      8014.80
```

for which we can also add the group variable:

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

Finally we will remove observation with missing glucagon values:

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

2 Descriptive statistics

2.1 Summary statistics

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

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

```
   outcome time observed missing   mean     sd    min     q1 median     q3    max
1   weight  B3m       20       0 128.97 20.269 100.90 115.30 123.10 139.82 173.00
2           B1w       20       0 121.24 18.910  95.70 107.78 114.50 134.53 162.20
3           A1w       20       0 115.70 18.275  89.90 102.22 110.60 128.38 155.00
4           A3m       20       0 102.36 17.054  78.80  90.40  98.50 108.25 148.00
5 glucagon  B3m       20       0   4.51  0.641   3.61   4.06   4.33   4.93   6.03
6           B1w       19       1   4.39  0.558   3.58   4.05   4.23   4.55   5.95
7           A1w       19       1   6.06  1.044   4.52   5.30   5.94   6.62   8.27
8           A3m       20       0   5.06  0.760   3.95   4.52   5.03   5.27   7.12
```

Correlation matrices are also ouput when a cluster and ordering variable have been specified (here respectively `id` and `time`):

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

```
  time observed missing mean   sd   min    q1 median  q3 max
1  B3m       20       0  129 20.3 100.9 115.3  123.1 140 173
2  B1w       20       0  121 18.9  95.7 107.8  114.5 135 162
3  A1w       20       0  116 18.3  89.9 102.2  110.6 128 155
4  A3m       20       0  102 17.1  78.8  90.4   98.5 108 148

 Pearson's correlation:
      B3m   B1w   A1w   A3m
B3m 1.000 0.990 0.986 0.946
B1w 0.990 1.000 0.997 0.959
A1w 0.986 0.997 1.000 0.966
A3m 0.946 0.959 0.966 1.000
```

Alternatively, the `partialCor` function can be used to compute correlation from the wide format, e.g.:

```
partialCor(weight1 + weight4 ~ 1, data = gastricbypassW)
```

```
                     estimate    se   df lower upper  p.value
rho(weight1,weight4)    0.946 0.105 31.1 0.867 0.978 8.46e-09
```

Partial correlations can be also computed, e.g.:

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

```
                     estimate    se   df lower upper  p.value
rho(weight1,weight4)    0.946 0.109 19.6 0.859  0.98 3.62e-07
```

The `partialCor` function can also be used to obtain group-specific correlations:

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

```
                          estimate    se   df  lower   upper p.value
0: rho(weight,glucagonAUC)   -0.281 0.148 21.1 -0.552  0.0442  0.0858
1: rho(weight,glucagonAUC)   -0.336 0.144 22.2 -0.594 -0.0156  0.0410
```

A p-value for the difference can be obtained 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)   -0.055 NA NA    NA    NA   0.789
```

2.2 Missing data patterns

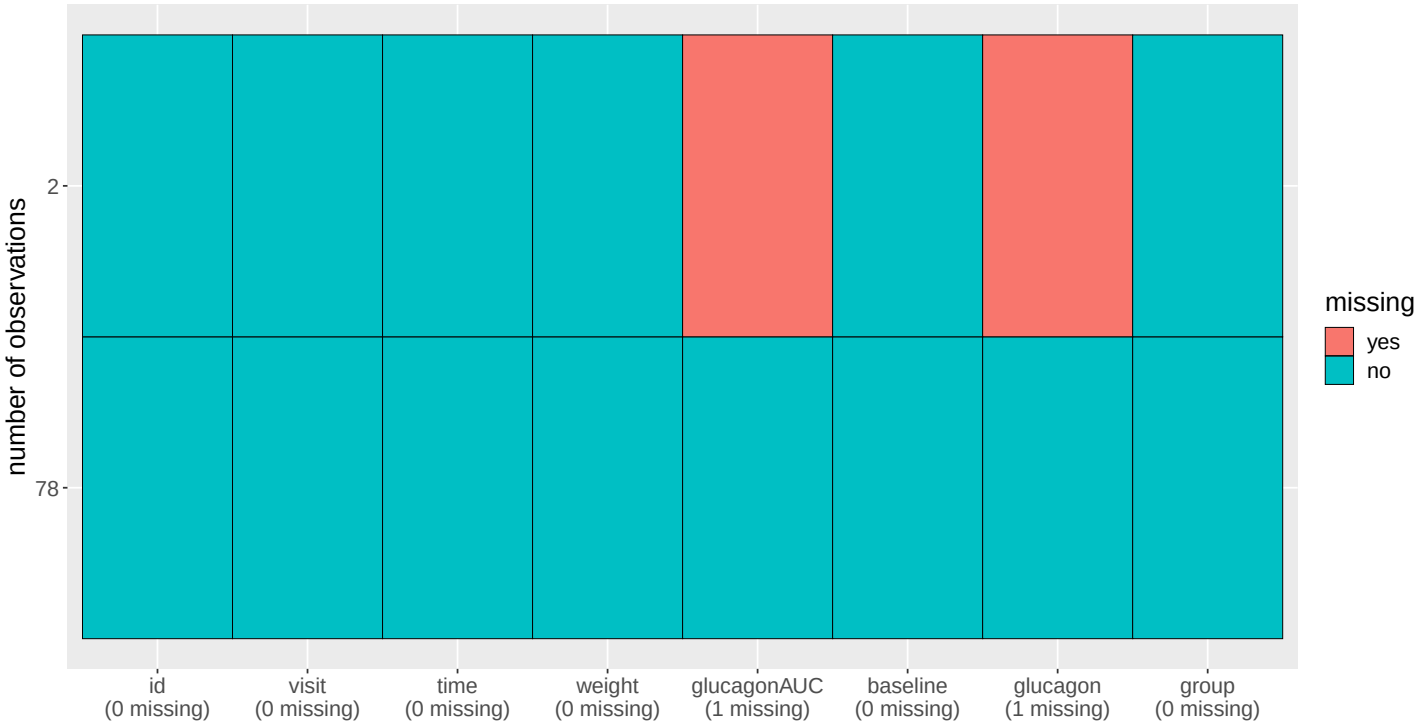
The `summarizeNA` identify the possible combinations of observed/missing data:

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

```
frequency missing.pattern n.missing id visit time weight glucagonAUC baseline glucagon group
       78        00000000         0  0     0    0      0           0        0        0     0
        2        00001010         2  0     0    0      0           1        0        1     0
```

A graphical representation can be obtained using `plot`:

```
plot(mp)
```


number of observations
2
78
id
(0 missing)
visit
(0 missing)
time
(0 missing)
weight
(0 missing)
glucagonAUC
(1 missing)
baseline
(0 missing)
glucagon
(1 missing)
group
(0 missing)
missing
yes
no

3 Linear mixed model

3.1 Classical covariance patterns

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

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

```
               Linear regression

 outcome/cluster/time: weight/id/time
 data                : 78 observations and distributed in 20 clusters
 parameter           : 5 mean ((Intercept) timeB1w timeA1w timeA3m glucagon)
                       1 variance (sigma)
 log-restr.likelihood: -323.086426918519
 convergence         : TRUE (0 iterations)
 covariance structure:
         B3m      B1w      A1w      A3m
B3m 330.0427   0.0000   0.0000   0.0000
B1w   0.0000 330.0427   0.0000   0.0000
A1w   0.0000   0.0000 330.0427   0.0000
A3m   0.0000   0.0000   0.0000 330.0427
```

and the **independence** structure:

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

```
               Linear regression with heterogeneous residual variance

 outcome/cluster/time: weight/id/time
 data                : 78 observations and distributed in 20 clusters
 parameter           : 5 mean ((Intercept) timeB1w timeA1w timeA3m glucagon)
                       4 variance (sigma k.B1w k.A1w k.A3m)
 log-restr.likelihood: -321.457830361849
 convergence         : TRUE (8 iterations)
 covariance structure:
         B3m      B1w      A1w      A3m
B3m 442.6475   0.0000   0.0000   0.0000
B1w   0.0000 418.9934   0.0000   0.0000
A1w   0.0000   0.0000 222.8463   0.0000
A3m   0.0000   0.0000   0.0000 237.2049
```

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

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

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

 outcome/cluster/time: weight/id/time
 data                : 78 observations and distributed in 20 clusters
 parameter           : 5 mean ((Intercept) timeB1w timeA1w timeA3m glucagon)
                       1 variance (sigma)
                       1 correlation (rho)
 log-restr.likelihood: -243.600523870252
 convergence         : TRUE (9 iterations)
 covariance structure:
         B3m      B1w      A1w      A3m
B3m 355.3062 344.6236 344.6236 344.6236
B1w 344.6236 355.3062 344.6236 344.6236
A1w 344.6236 344.6236 355.3062 344.6236
A3m 344.6236 344.6236 344.6236 355.3062
```

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

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

```
               Linear Mixed Model with a Toeplitz covariance matrix

 outcome/cluster/time: weight/id/time
 data                : 78 observations and distributed in 20 clusters
 parameter           : 8 mean ((Intercept) timeB1w timeA1w timeA3m group timeB1w:group timeA1w:grou
                       4 variance (sigma k.B1w k.A1w k.A3m)
                       3 correlation (rho(B3m,B1w) rho(B3m,A1w) rho(B3m,A3m))
 log-restr.likelihood: -221.152940926053
 convergence         : TRUE (21 iterations)
 correlation structure:
          B3m       B1w       A1w       A3m
B3m 1.0000000 0.9854133 0.9676223 0.9489216
B1w 0.9854133 1.0000000 0.9854133 0.9676223
A1w 0.9676223 0.9854133 1.0000000 0.9854133
A3m 0.9489216 0.9676223 0.9854133 1.0000000
```

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

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

```
               Linear Mixed Model with an unstructured covariance matrix

 outcome/cluster/time: weight/id/time
 data                : 78 observations and distributed in 20 clusters
 parameter           : 5 mean ((Intercept) timeB1w timeA1w timeA3m glucagon)
                       4 variance (sigma k.B1w k.A1w k.A3m)
                       6 correlation (rho(B3m,B1w) rho(B3m,A1w) rho(B3m,A3m) rho(B1w,A1w) rho(B1w,A
 log-restr.likelihood: -216.318937004306
 convergence         : TRUE (22 iterations)
 covariance structure:
         B3m      B1w      A1w      A3m
B3m 411.3114 381.9734 352.6400 318.8573
B1w 381.9734 362.7326 335.4649 304.6314
A1w 352.6400 335.4649 311.6921 285.8077
A3m 318.8573 304.6314 285.8077 280.9323
```

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

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

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

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

outcome/cluster/time: weight/id/time
data                : 78 observations and distributed in 20 clusters
parameter           : 8 mean ((Intercept) timeB1w timeA1w timeA3m group timeB1w:group timeA1w:group
                      2 variance (sigma:0 sigma:1)
                      2 correlation (rho:0 rho:1)
log-restr.likelihood: -229.203435252784
convergence         : TRUE (6 iterations)
```

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

```
eSUN.lmm <- lmm(weight ~ time*group + glucagon,
                repetition = ~time|id, structure = UN(~group),
                data = dfL)
eSUN.lmm
```

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

outcome/cluster/time: weight/id/time
data                : 78 observations and distributed in 20 clusters
parameter           : 9 mean ((Intercept) timeB1w timeA1w timeA3m group glucagon timeB1w:group time
                      8 variance (sigma:0 sigma:1 k.B1w:0 k.A1w:0 k.A3m:0 k.B1w:1 k.A1w:1 k.A3m:1)
                      12 correlation (rho(B3m,B1w):0 rho(B3m,A1w):0 rho(B3m,A3m):0 rho(B1w,A1w):0 r
log-restr.likelihood: -197.171312062213
convergence         : TRUE (50 iterations)
```

with covariance structure:

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

```
$`0`
         B3m      B1w      A1w      A3m
B3m 348.0783 334.7404 334.7404 334.7404
B1w 334.7404 348.0783 334.7404 334.7404
A1w 334.7404 334.7404 348.0783 334.7404
A3m 334.7404 334.7404 334.7404 348.0783

$`1`
         B3m      B1w      A1w      A3m
B3m 345.5863 340.1538 340.1538 340.1538
B1w 340.1538 345.5863 340.1538 340.1538
A1w 340.1538 340.1538 345.5863 340.1538
A3m 340.1538 340.1538 340.1538 345.5863
```

```
$`0`
         B3m      B1w      A1w      A3m
B3m 417.3374 382.8829 362.5674 301.7430
B1w 382.8829 364.4515 346.4039 292.7507
A1w 362.5674 346.4039 331.1789 282.9301
A3m 301.7430 292.7507 282.9301 253.3324

$`1`
         B3m      B1w      A1w      A3m
B3m 383.8877 363.6405 336.5771 350.0416
B1w 363.6405 347.9898 321.5908 331.5182
A1w 336.5771 321.5908 297.5329 308.1345
A3m 350.0416 331.5182 308.1345 334.8267
```

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(weight ~ time*group,repetition = ~time|id,
                structure = CS(~baseline, heterogeneous = FALSE), data = dfL)
eBCS.lmm
cat(" covariance structure: \n");sigma(eBCS.lmm)
```

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

 outcome/cluster/time: weight/id/time
 data                : 78 observations and distributed in 20 clusters
 parameter           : 8 mean ((Intercept) timeB1w timeA1w timeA3m group timeB1w:group timeA1w:grou
                       1 variance (sigma)
                       2 correlation (rho(TRUE) rho(TRUE,FALSE))
 log-restr.likelihood: -230.532819632968
 convergence         : TRUE (6 iterations)
 covariance structure:
         B3m      B1w      A1w      A3m
B3m 346.7441 339.3256 336.1825 336.1825
B1w 339.3256 346.7441 336.1825 336.1825
A1w 336.1825 336.1825 346.7441 339.3256
A3m 336.1825 336.1825 339.3256 346.7441
```

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

```
eBUN.lmm <- lmm(weight ~ time*group, repetition = ~time|id,
                structure = CS(~baseline, heterogeneous = TRUE), data = dfL)
eBUN.lmm
cat(" covariance structure: \n");sigma(eBUN.lmm)
```

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

 outcome/cluster/time: weight/id/time
 data                : 78 observations and distributed in 20 clusters
 parameter           : 8 mean ((Intercept) timeB1w timeA1w timeA3m group timeB1w:group timeA1w:grou
                       2 variance (sigma k.TRUE)
                       3 correlation (rho(TRUE) rho(TRUE,FALSE) rho(FALSE))
 log-restr.likelihood: -227.461008305704
 convergence         : TRUE (6 iterations)
 covariance structure:
         B3m      B1w      A1w      A3m
B3m 378.0328 372.8100 336.3064 336.3064
B1w 372.8100 378.0328 336.3064 336.3064
A1w 336.3064 336.3064 315.6358 306.0647
A3m 336.3064 336.3064 306.0647 315.6358
```

[1] similar to nested random effects

3.2 User-specific covariance patterns

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

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

where:

- $\boldsymbol{\sigma} = f(\boldsymbol{\theta}_\sigma, Z_\sigma)$ 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,time,...){
  n.time <- length(time)
  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),
                  time = 1:6)
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)
n <- 1000
Y <- rmvnorm(n, mean = rep(0,6), sigma = Rho)
dfL2 <- reshape2::melt(cbind(id = 1:n, as.data.frame(Y)), id.vars = "id")
dfL2$time  <- dfL2$variable
dfL2 <- dfL2[order(dfL2$id),]
dfL2[1:8,]
```

```
     id variable      value time
1     1       V1 -0.9842079   V1
1001  1       V2 -0.3681245   V2
2001  1       V3 -1.6174652   V3
3001  1       V4 -1.4994103   V4
4001  1       V5  0.7493107   V5
5001  1       V6 -1.0719657   V6
2     2       V1  1.2402726   V1
1002  2       V2  0.6494215   V2
```

To fit the corresponding mixed model, we first define a specific covariance structure using the `CUSTOM` function:

```
myStruct <- CUSTOM(~variable,
                   FCT.sigma = function(p,time,X){rep(p,length(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=dfL2,
                   df = FALSE) ## df = FALSE to save computation time
logLik(e.lmmCUSTOM)
```

```
[1] -7962.243
```

The optimization procedure is not very fast but 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
```

Note that specifying a classical structure (e.g. compound symmetry):

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

via CUSTOM:

```
logLik(lmm(value~time,
           repetition = ~time|id,
           structure = myCS,
           data = dfL2, df = FALSE
           ))
```

```
[1] -8186.859
```

will be the same as using the pre-specified structure (up the certain user-friendly displays):

```
logLik(lmm(value~time,
           repetition = ~time|id,
           structure = "CS",
           data = dfL2, df = FALSE))
```

```
[1] -8186.859
```

3.3 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)     timeB1w     timeA1w     timeA3m    glucagon       sigma         rho
159.1349871  -7.7137607  -2.3202963 -22.9747234  -6.6820191  18.1670760   0.8960476

Loop:
*********
(Intercept)     timeB1w     timeA1w     timeA3m    glucagon       sigma         rho
125.2601602  -7.6194918 -14.4951323 -27.0514694   0.8217879  18.8495686   0.9699341
Convergence after 9 iterations: max score=3.680809e-06 | max change in coefficient=1.877273e-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 = 2))
```

```
Initialization:
(Intercept)     timeB1w     timeA1w     timeA3m    glucagon       sigma         rho
125.2601602  -7.6194918 -14.4951323 -27.0514694   0.8217879  18.8495686   0.9699341

Loop:

(Intercept)     timeB1w     timeA1w     timeA3m    glucagon       sigma         rho
125.2601602  -7.6194918 -14.4951323 -27.0514694   0.8217879  18.8495686   0.9699341
Convergence after 0 iteration: max score=3.680809e-06
```

- 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)     timeB1w     timeA1w     timeA3m    glucagon       sigma         rho
125.2601602  -7.6194918 -14.4951323 -27.0514694   0.8217879  19.0853249   0.9737808

Loop:
*****
(Intercept)     timeB1w     timeA1w     timeA3m    glucagon       sigma         rho
125.2601601  -7.6194918 -14.4951323 -27.0514694   0.8217879  18.8495690   0.9699341
Convergence after 5 iterations: max score=2.820304e-06 | max change in coefficient=4.171839e-06
```

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

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

```
Initialization:
  (Intercept)       timeB1w       timeA1w       timeA3m         group timeB1w:group
  134.2700000    -8.2800000   -14.1100000   -29.6100000   -10.6000000     1.0505605
timeA1w:group timeA3m:group       sigma:0       sigma:1         rho:0         rho:1
    1.7562258     6.0100000    18.8495686    18.8495686     0.9699341     0.9699341

Loop:
*******
  (Intercept)       timeB1w       timeA1w       timeA3m         group timeB1w:group
  134.2700000    -8.2800000   -14.1100000   -29.6100000   -10.6000000     1.0444208
timeA1w:group timeA3m:group       sigma:0       sigma:1         rho:0         rho:1
    1.7525468     6.0100000    18.6568561    18.5899521     0.9616812     0.9842804
Convergence after 7 iterations: max score=2.208526e-05 | max change in coefficient=6.343994e-06
```

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 u.pdated 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.

3.4 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: dfL

  - 20 clusters
  - 78 observations
  - between 3 and 4 observations per cluster

Summary of the outcome and covariates:

    $ weight  : num  127 165 110 146 113 ...
    $ time    : Factor w/ 4 levels "B3m","B1w","A1w",..: 1 1 1 1 1 1 1 1 1 1 ...
    $ glucagon: num  4.03 5.24 4.93 4.32 4.38 ...
    reference level: time=B3m

Estimation procedure

  - Restricted Maximum Likelihood (REML)
  - log-likelihood :-216.3189
  - parameters: mean = 5, variance = 4, correlation = 6
  - convergence: TRUE (22 iterations)
    largest |score| = 7.034659e-05 for k.A1w
            |change|= 1.09738491005373e-06 for (Intercept)

Residual variance-covariance: unstructured

  - correlation structure: ~time - 1
         B3m   B1w   A1w   A3m
    B3m 1.000 0.989 0.985 0.938
    B1w 0.989 1.000 0.998 0.954
    A1w 0.985 0.998 1.000 0.966
    A3m 0.938 0.954 0.966 1.000

  - variance structure: ~time
              standard.deviation ratio
    sigma.B3m               20.3 1.000
    sigma.B1w               19.0 0.939
    sigma.A1w               17.7 0.871
    sigma.A3m               16.8 0.826
```

```
Fixed effects: weight ~ time + glucagon

               estimate    se   df   lower   upper  p.value
    (Intercept)   132.98 4.664 19.8 123.243 142.717  < 2e-16 ***
    timeB1w        -7.882 0.713 19.2  -9.374   -6.39 9.27e-10 ***
    timeA1w       -11.788 1.018 21.6   -13.9  -9.676 9.55e-11 ***
    timeA3m       -26.122 1.656 18.8 -29.591 -22.654 2.62e-12 ***
    glucagon       -0.888 0.242 13.7  -1.408  -0.369  0.00257  **
    ---------------------------------------------------------
   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.

3.5 Extract estimated coefficients

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

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

```
(Intercept)     timeB1w     timeA1w     timeA3m    glucagon
132.9801355  -7.8822331 -11.7879545 -26.1223908  -0.8883081
```

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

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

```
     sigma      k.B1w      k.A1w      k.A3m
20.2808131  0.9390916  0.8705176  0.8264480
```

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

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

```
sigma.B3m sigma.B1w sigma.A1w sigma.A3m
 20.28081  19.04554  17.65480  16.76104
```

The marginal means at each timepoint can be obtained using `dummy.coef`:

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

```
  time estimate       se       df     lower    upper
1  B3m 128.5386 4.536445 18.97584 119.04289 138.0343
2  B1w 120.6564 4.261691 19.04078 111.73783 129.5749
3  A1w 116.7506 3.956964 19.04925 108.47007 125.0312
4  A3m 102.4162 3.747908 19.05531  94.57328 110.2591
```

3.6 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) 132.9801355 4.6642475 19.75815 123.243045 142.7172256 0.000000e+00
timeB1w      -7.8822331 0.7131797 19.17147  -9.374032  -6.3904339 9.273644e-10
timeA1w     -11.7879545 1.0175135 21.64404 -13.900162  -9.6757467 9.552470e-11
timeA3m     -26.1223908 1.6564077 18.84049 -29.591280 -22.6535021 2.617462e-12
glucagon     -0.8883081 0.2416081 13.70759  -1.407545  -0.3690712 2.571605e-03
```

Values for the all correlation parameters can be displayed too, by specifying `effect="all"`:

```
model.tables(eUN.lmm, effect = "all")
```

```
                estimate           se       df       lower       upper      p.value
(Intercept)  132.9801355 4.664247e+00 19.75815 123.2430454 142.7172256 0.000000e+00
timeB1w       -7.8822331 7.131797e-01 19.17147  -9.3740323  -6.3904339 9.273644e-10
timeA1w      -11.7879545 1.017513e+00 21.64404 -13.9001622  -9.6757467 9.552470e-11
timeA3m      -26.1223908 1.656408e+00 18.84049 -29.5912795 -22.6535021 2.617462e-12
glucagon      -0.8883081 2.416081e-01 13.70759  -1.4075449  -0.3690712 2.571605e-03
sigma         20.2808131 1.042207e+08 17.94875  14.4225149  28.5187002           NA
k.B1w          0.9390916 8.746246e-02 19.25090   0.8742815   1.0087060 8.159292e-02
k.A1w          0.8705176 9.733113e-02 20.32066   0.7996375   0.9476805 2.778018e-03
k.A3m          0.8264480 1.820402e-01 19.48030   0.6997216   0.9761257 2.692889e-02
rho(B3m,B1w)   0.9889048 9.815766e-02 32.79091   0.9719687   0.9956310 7.778223e-13
rho(B3m,A1w)   0.9848800 9.911546e-02 26.28819   0.9614535   0.9941119 5.780221e-11
rho(B3m,A3m)   0.9380157 1.061121e-01 23.56848   0.8470249   0.9755995 1.153943e-07
rho(B1w,A1w)   0.9976791 9.925175e-02 27.01628   0.9939113   0.9991163 3.730349e-14
rho(B1w,A3m)   0.9542904 1.035349e-01 24.72225   0.8860968   0.9820453 1.782701e-08
rho(A1w,A3m)   0.9658511 1.015050e-01 27.88668   0.9147964   0.9865286 1.450022e-09
```

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) 132.9801355 0.000000e+00
timeB1w      -7.8822331 9.273644e-10
timeA1w     -11.7879545 9.552470e-11
timeA3m     -26.1223908 2.617462e-12
glucagon     -0.8883081 2.571605e-03
```

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

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) 132.9801355 4.6642475  28.510523 19.75815 123.243045 142.7172256 0.000000e+00
timeB1w      -7.8822331 0.7131797 -11.052240 19.17147  -9.374032  -6.3904339 9.273644e-10
timeA1w     -11.7879545 1.0175135 -11.585060 21.64404 -13.900162  -9.6757467 9.552470e-11
timeA3m     -26.1223908 1.6564077 -15.770508 18.84049 -29.591280 -22.6535021 2.617462e-12
glucagon     -0.8883081 0.2416081  -3.676648 13.70759  -1.407545  -0.3690712 2.571605e-03
```

3.7 Extract estimated residual variance-covariance structure

The method `sigma` can be used to output the modeled residual covariance structure:

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

```
         B3m      B1w      A1w      A3m
B3m 411.3114 381.9734 352.6400 318.8573
B1w 381.9734 362.7326 335.4649 304.6314
A1w 352.6400 335.4649 311.6921 285.8077
A3m 318.8573 304.6314 285.8077 280.9323
```

and then converted to a correlation matrix using `cov2cor`:

```
cov2cor(Sigma)
```

```
          B3m       B1w       A1w       A3m
B3m 1.0000000 0.9889048 0.9848800 0.9380157
B1w 0.9889048 1.0000000 0.9976791 0.9542904
A1w 0.9848800 0.9976791 1.0000000 0.9658511
A3m 0.9380157 0.9542904 0.9658511 1.0000000
```

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

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

```
         B3m      A1w      A3m
B3m 411.3114 352.6400 318.8573
A1w 352.6400 311.6921 285.8077
A3m 318.8573 285.8077 280.9323
```

or for a new individual:

```
newdata <- data.frame(id = "X", time = c("B3m","B1w","A1w","A3m"))
sigma(eUN.lmm, cluster = newdata)
```

```
         B3m      B1w      A1w      A3m
B3m 411.3114 381.9734 352.6400 318.8573
B1w 381.9734 362.7326 335.4649 304.6314
A1w 352.6400 335.4649 311.6921 285.8077
A3m 318.8573 304.6314 285.8077 280.9323
```

3.8 Random effects

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

$$\Omega = Z\Omega_1 Z^\intercal + \Omega_2$$

where:

- Z is the design matrix associated to the possibly nested clustering factors
- Ω_1 is the variance-covariance of the random effects
- Ω_2 the residual-variance covariance conditional to the random effects.

The joint distribution between the outcome $\boldsymbol{Y}$ and the random effects $\boldsymbol{\eta}$ is

$$\begin{bmatrix} \boldsymbol{Y} \\ \boldsymbol{\eta} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \Omega & Z\Omega_1 \\ \Omega_1 Z^\intercal & \Omega_1 \end{bmatrix} \right)$$

Denoting by $\varepsilon_i = \boldsymbol{Y}_i - \boldsymbol{\mu}_i$ the vector of marginal residuals relative to individual i with variance-covariance matrix Ω_i, the j-th random effect is the expected value given the residual:

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

where ω_{1j} the variance of the random effect. This is what the `coef` method returns when setting the argument `effects` to `"ranef"`:

```
head(coef(eCS.lmm, effects = "ranef"))
```

```
          id
1  0.9036038
2 32.5542378
3 -18.3099658
4 20.2561307
5 -15.4258816
6 19.3751847
```

```
head(coef(eBCS.lmm, effects = "ranef"))
```

```
          id   baseline1  baseline2
1  4.931442  0.52901983 -0.4829138
2 28.390660 -0.09204109  0.3574766
3 -13.728389  0.18951039 -0.3178625
4 15.645550  0.82309894 -0.6768225
5 -11.246852 -0.30658155  0.2014303
6 15.002108 -2.64303027  2.7832909
```

3.9 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. $\Omega = Z\Omega_1 Z^\intercal + \omega I$ where I is the identity matrix and ω 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 variance ω:

```
sigma2 <- coef(eCS.lmm, effect = "variance")^2
tau <- coef(eCS.lmm, effect = "correlation")*sigma2
omega <- 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 * omega
c(df.res = df.res, SSE = SSE)
```

```
 df.res      SSE
73.0000 779.8304
```

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
```

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.glucagon <- 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 * omega)
SSR.glucagon <- as.double(SSRstar.glucagon * omega)
c(time = SSR.time, glucagon = SSR.glucagon)
```

```
      time   glucagon
6986.78351   18.83074
```

3.10 Proportion of explained variance and partial correlation

⚠ The definition of explained variance is not straightforward with mixed models. Intuitively considering the variance across several outcomes will be hard to interpret unless all outcomes have the same variance. Similar consideration holds for partial correlation. This is why LMMstar does not output these quantities by default. Nevertheless for specific covariance structure, namely independence and compound symmetry (with positive correlation) structure, explained variance and partial correlation can be deduced from the `lmm` object. Importantly, with these structures the residuals can be reparametrised as random effects plus independent residuals, i.e. $\Omega = Z\Omega_1 Z^\intercal + \omega I$ where I is the identity matrix and ω the variance of these independent residuals.

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.glucagon/ (SSR.glucagon + SSE))
```

```
[1] 0.89959197 0.02357789
```

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

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

```
             Partial correlation

          estimate    se   df  lower  upper  p.value
timeB1w     -0.646 0.055 18.6 -0.762  -0.53 5.11e-10
timeA1w     -0.765 0.035  9.5 -0.845 -0.686 2.07e-09
```

```
timeA3m    -0.946 0.006  2.4 -0.969 -0.923 6.80e-06
glucagon    0.154 0.114 45.3 -0.076  0.383    0.184
-----------------------------------------------
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           0.9 0.011  2.4  0.857 0.942 4.09e-05
glucagon   0.024 0.035 45.3 -0.047 0.094    0.503
global     0.906 0.011  2.3  0.866 0.946 4.51e-05
-----------------------------------------------
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 covariance structures mentioned in this section is questionnable.

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

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

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)$multivariate
setNames(aCS.aov$statistic/(aCS.aov$statistic+aCS.aov$df.denom), aCS.aov$test)
```

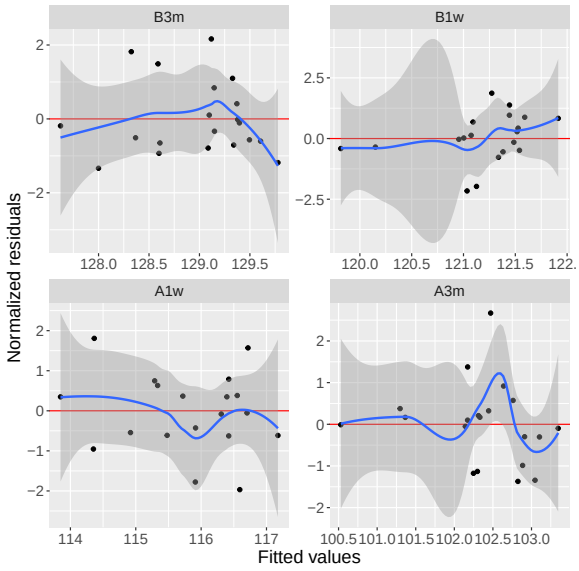
```
      time   glucagon
0.80163957 0.03162017
```

3.11 Model diagnostic

The method `plot` can be used to display diagnostic plots about:

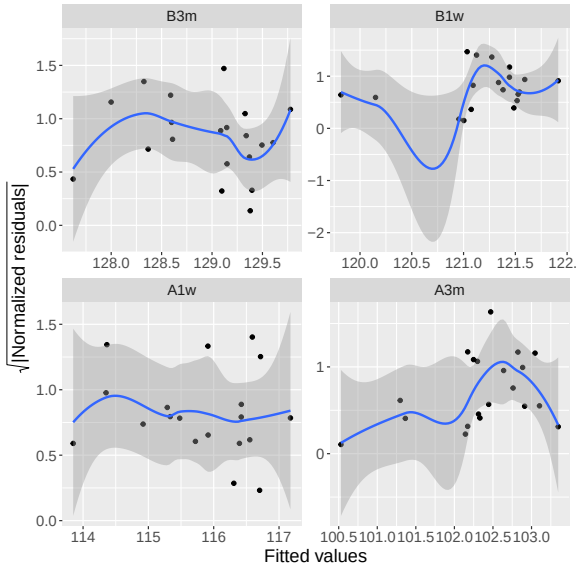
- misspecification of the mean structure

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



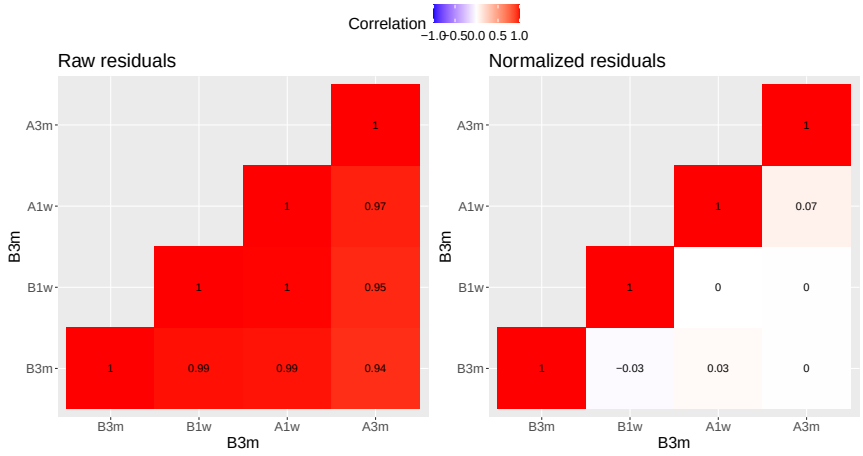
- misspecification of the variance structure

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



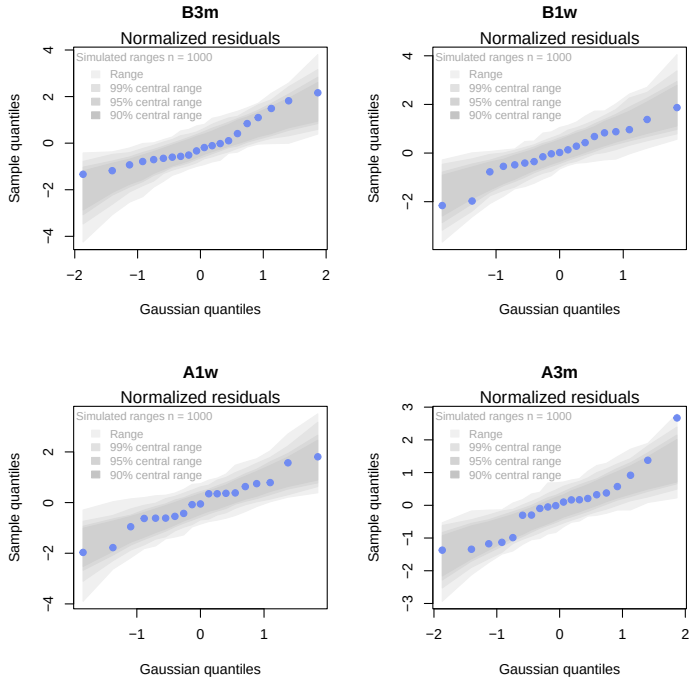
- misspecification of the correlation structure

```
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.qqplot = "qqtest")
## 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.

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

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

```
  cluster      r.B3m      r.B1w      r.A1w       r.A3m
1       1 -0.1082872  0.4283943  0.7477306  0.91794015
2       2  1.8182348 -0.3516996  1.5698307 -0.98743171
3       3 -0.9318737 -0.7728221  0.6315751  0.16549699
4       4  0.8408969  1.8695564  0.3485784 -0.09662565
5       5 -0.7882340         NA -0.6128276  0.09933842
6       6  1.4896141 -1.9727358 -1.9672939 -1.37068983
```

or in the long format:

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

```
[1] -0.1082872  1.8182348 -0.9318737  0.8408969 -0.7882340  1.4896141
```

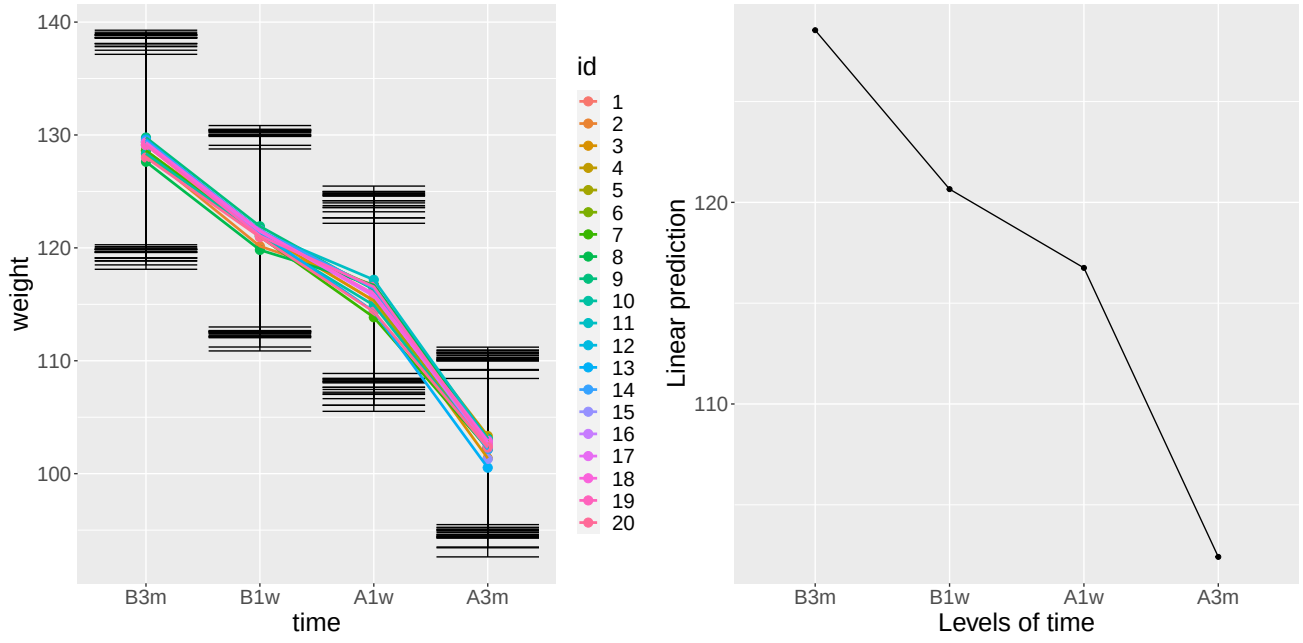
Various type of residuals can be extract but the normalized one are recommanded when doing model checking.

3.12 Model fit

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

```
library(ggplot2) ## left panel
plot(eUN.lmm, type = "fit", color = "id", ci.alpha = NA, size.text = 20)
```

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

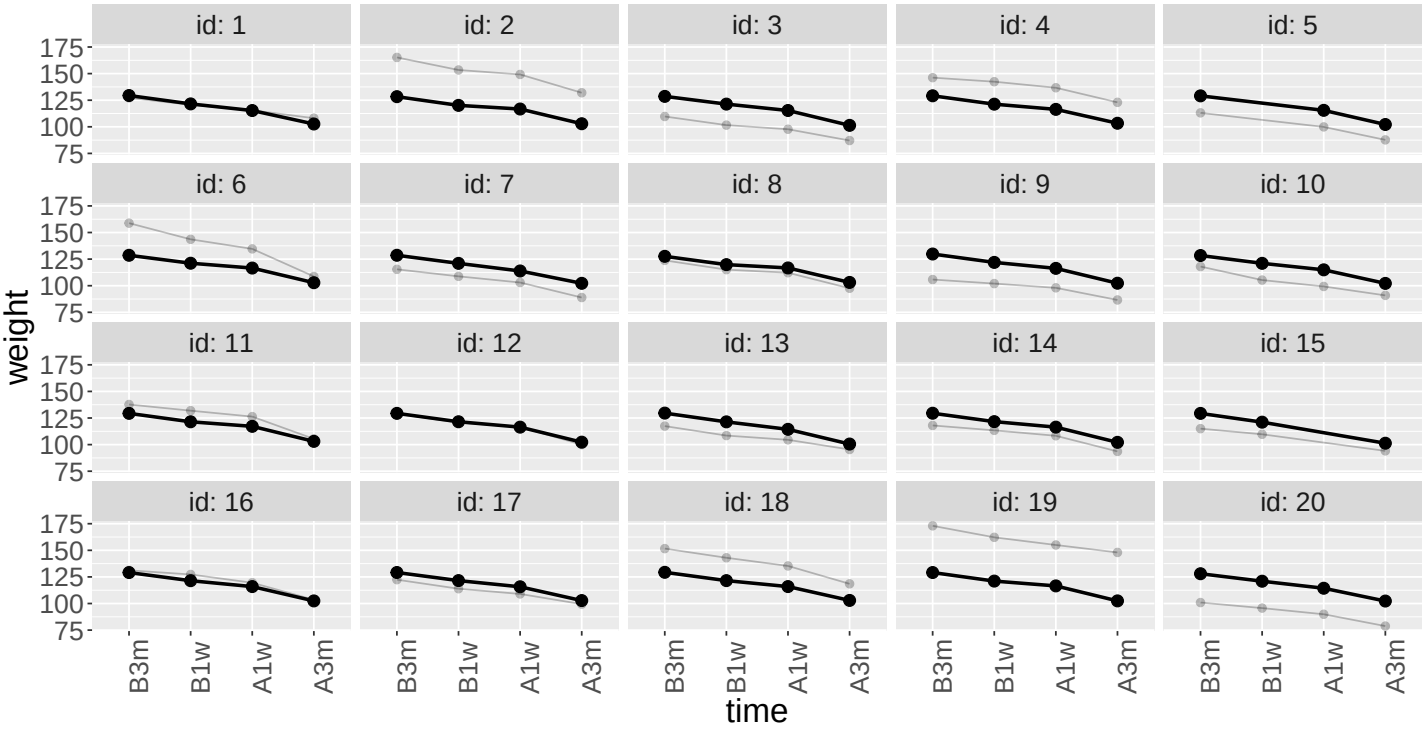


In the first case each possible curve is displayed while in the latter the average curve (over glucagon values). With the `plot` method, it is possible to display a curve specific to a glucagon value via the argument `at`:

```
plot(eUN.lmm, type = "fit", at = data.frame(glucagon = 10), color = "glucagon")
## result not shown
```

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:

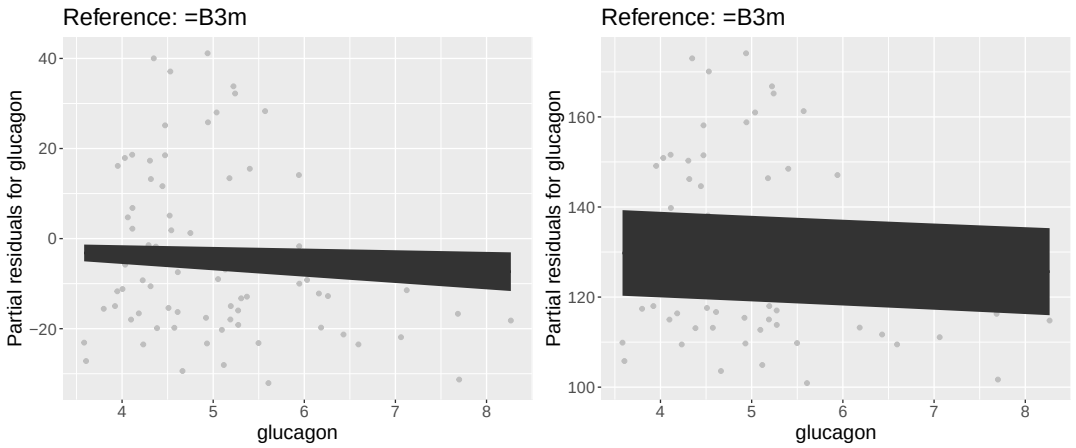
```
gg <- autoplot(eUN.lmm, type = "fit", obs.alpha = 0.25, ci = FALSE)$plot
gg <- gg + facet_wrap(~id, labeller = label_both)
gg <- gg + theme(axis.text.x=element_text(angle = 90, hjust = 0))
gg
```



3.13 Partial residuals

Partial residuals can also be displayed via the `plot` method:

```
gg1 <- autoplot(eUN.lmm, type = "partial", var = "glucagon")$plot
gg2 <- autoplot(eUN.lmm, type = "partial", var = c("(Intercept)","glucagon"))$plot
ggarrange(gg1,gg2)
```



Their value can be extracted via the `residuals` method, e.g.:

```
df.pres <- residuals(eUN.lmm, type = "partial", var = "glucagon", keep.data = TRUE)
head(df.pres)
```

```
  id visit time weight glucagonAUC baseline glucagon group  r.partial
1  1     1  B3m  127.2     5032.50     TRUE 4.034616     1  -5.780135
2  2     1  B3m  165.2    12142.50     TRUE 5.240766     0  32.219865
3  3     1  B3m  109.7    10321.35     TRUE 4.931824     1 -23.280135
4  4     1  B3m  146.2     6693.00     TRUE 4.316306     0  13.219865
5  5     1  B3m  113.1     7090.50     TRUE 4.383738     1 -19.880135
6  6     1  B3m  158.8    10386.00     TRUE 4.942791     0  25.819865
```

This matches manual calculation:

```
m.pres <- dfL$weight - model.matrix(~time,dfL) %*% coef(eUN.lmm)[1:4]
range(df.pres$r.partial - m.pres, na.rm = TRUE)
```

```
[1] -1.065814e-14  1.421085e-14
```

Note: to match the partial residuals obtained from `lm`:

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

one should use `type` equal to `"partial-center"` which also removes the average effect of the covariate:

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

```
[1] -6.883383e-15  8.881784e-15
```

3.14 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  -5.313  -2.498   2e-05 ***
   timeA3m - timeB1w   -18.24 1.323   19 -21.372 -15.109  <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   -9.83  -5.935  <1e-05 ***
   A1w - B3m  -11.788 1.018 21.6 -14.567  -9.009  <1e-05 ***
   A3m - B3m  -26.122 1.656 18.8 -30.646 -21.599  <1e-05 ***
   A1w - B1w   -3.906 0.595 17.9   -5.53  -2.282   4e-05 ***
   A3m - B1w   -18.24 1.323   19 -21.853 -14.627  <1e-05 ***
   A3m - A1w  -14.334 1.057 20.3  -17.22 -11.449  <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"),
                anova(eCS.lmm, effects = "glucagon = 0"),
                anova(eUN.lmm, effects = "glucagon = 0"),
                name = c("Ind","CS","UN"))
summary(Manova)
```

```
               Multivariate Wald test

         Chi2-statistic      df  p.value
  all: 1          6.393 (3,Inf) 0.000251 ***
  ------------------------------------------
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 2.574 34.2 -14.848 -1.692  0.0116 *
  CS: glucagon      0.822  0.59 53.8  -0.687   2.33  0.4321
  UN: glucagon     -0.888 0.353 13.7   -1.79  0.014  0.0546 .
  ------------------------------------------------------------
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)
Robust standard errors are derived from the observed information (column se).
```

3.15 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:

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

```
gastricbypassW <- reshape(dfL[,c("id","time","weight","group")],
                          direction = "wide",
                          timevar = "time", idvar = c("id","group"))
e.ANCOVA <- lm(weight.A3m ~ weight.B3m + group, data = gastricbypassW)
summary(e.ANCOVA)$coef
```

```
             Estimate Std. Error    t value     Pr(>|t|)
(Intercept) -5.9285136 8.78006389 -0.6752244 5.086140e-01
weight.B3m   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.

3.16 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.
```

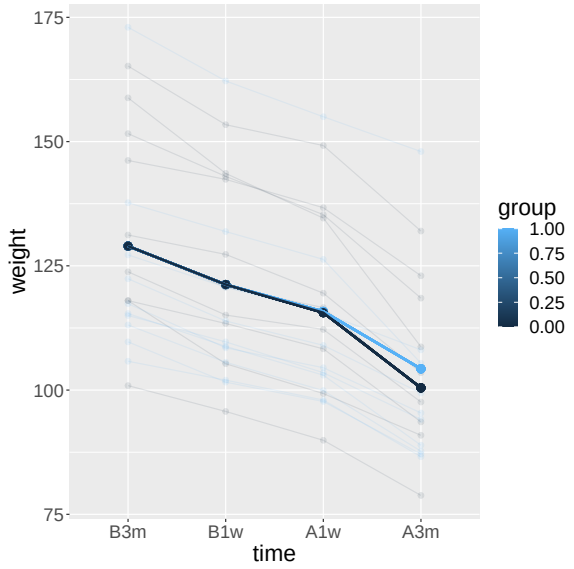
with the following coefficients:

```
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)
```



3.17 Marginal means

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

```
gastricbypassL$group2 <- as.numeric(gastricbypassL$id) %% 3 == 0
e.group <- lmm(glucagon ~ time*group2, data = gastricbypassL,
               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 <- cbind(gastricbypassL, predict(e.group, newdata = gastricbypassL))
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       20       0 4.390071 0.1617778 4.149209 4.149209 4.493298 4.493298 4.493298
3  A1w       20       0 6.044056 0.2109650 5.729961 5.729961 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  0.5065
 TRUE - FALSE A1w    -0.449 0.525 18.4   -1.852    0.955  -0.855  0.7960
 TRUE - FALSE A3m     0.312 0.374 18.0   -0.688    1.312   0.834  0.8085

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  0.9935
 treat20 - treat21 A3m     -3.83 2.107  18  -1.819  0.3019

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

3.18 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  NA        NA       NA
2  1  B1w       NA        0 125.09790 0.6362754 Inf 123.85083 126.3450
3  2  B3m 100.0000        0        NA        NA  NA        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
```

4 Equivalence between LMM and LM

4.1 Regressing 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)
```

In the case of missing data:

```
index.missing32 <- which(is.na(gastricbypassW$changeG32))
index.missing32
```

```
[1]  5 15
```

LMM uses a full information approach while a linear model perform a complete case approach [4]:

```
gastricbypassW$changeG32 <- gastricbypassW$glucagonAUC3-gastricbypassW$glucagonAUC2
e.change32 <- lm(changeG32 ~ weight1, data = gastricbypassW)
summary(e.change32)$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
```

which is the same as removing the two individuals with missing data:

```
coef(lm(changeG32 ~ weight1, data = gastricbypassW[-index.missing32,]))
```

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

The LMM would give (here slightly) different estimates:

```
gastricbypassL32 <- gastricbypassL[gastricbypassL$visit %in% c(3,2),]
gastricbypassL32$time <- droplevels(gastricbypassL32$time)
gastricbypassL32$weight1 <- gastricbypassW$weight1[gastricbypassL32$id]

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
```

These estimates can be retrived using a linear model where the mixed model has imputed the conditional expectation of the missing values given the observed value and the estimated model parameters:

```
gastricbypassWA <- fitted(e.lmm32, impute = TRUE, keep.newdata = TRUE, format = "wide")
gastricbypassWA$change32 <- gastricbypassWA$glucagonAUC.A1w - gastricbypassWA$glucagonAUC.B1w
coef(lm(change32 ~ weight1, data = gastricbypassWA))
```

[4] 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

```
(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.

4.2 Regressing the change against another change

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
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 as mixed model with all measurement as outcomes, and where the parameter of interest is

```
keep.col <- c("id","weight1","weight4","glucagonAUC1","glucagonAUC4")
gastricbypassL4 <- melt(gastricbypassW[,keep.col], id.vars = "id")

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

estimate(e.lmm4, function(p){
  sigma()
})
```

5 Missing values and imputation

5.1 Complete case approach

```
gastricbypassW$changeG32 <- gastricbypassW$glucagonAUC3-gastricbypassW$glucagonAUC2
e.change32 <- lm(changeG32 ~ weight1, data = gastricbypassW)
```

5.2 Full information approach

We now consider the glucagon level as an outcome. 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
```

Linear mixed model can handle missing value in the outcome variable, assuming that missigness is random conditional on the covariate and observed outcome values. The `lmm` function can be used "as usual":

```
eUN.lmmNA <- lmm(glucagon ~ time,
                 repetition = ~time|id, structure = "UN",
                 data = gastricbypassL)
summary(eUN.lmmNA)
```

```
                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:

    $ glucagon: num  4.03 5.24 4.93 4.32 4.38 ...
    $ time    : Factor w/ 4 levels "B3m","B1w","A1w",..: 1 1 1 1 1 1 1 1 1 1 ...
    reference level: time=B3m
```

The visible difference in the summary is when describing the dataset: we can see that some repetitions (here 2) have been ignored as the outcome was missing. So for some clusters only 3 values were analyzed instead of 4.

5.3 Imputation

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

```
fitted(eUN.lmmNA, impute = TRUE)
```

```
[1] 4.256984 6.497856
```

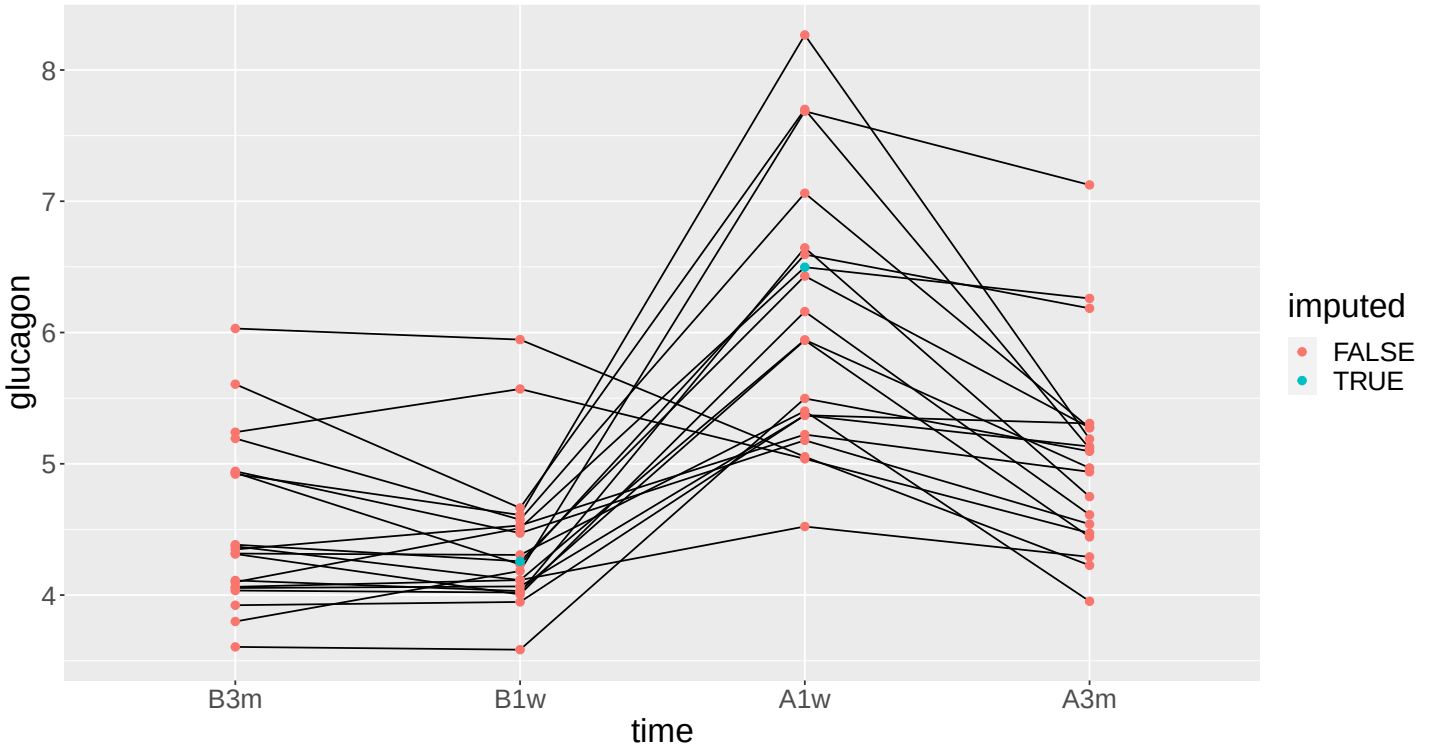
When using the argument `keep.newdata=TRUE`, the missing outcome value has been replaced by its most likely value (which is the same as the dynamic prediction, describedy previously):

```
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 group2 imputed
5   5     1  B3m  113.1      7090.5     TRUE 4.383738     1  FALSE   FALSE
15 15     1  B3m  115.0      5410.5     TRUE 4.098741     1   TRUE   FALSE
25  5     2  B1w  105.6          NA     TRUE 4.256984     1  FALSE    TRUE
35 15     2  B1w  109.7      7833.0     TRUE 4.509697     1   TRUE   FALSE
45  5     3  A1w   99.9     19155.0    FALSE 6.430376     1  FALSE   FALSE
55 15     3  A1w  103.5          NA    FALSE 6.497856     1   TRUE    TRUE
65  5     4  A3m   87.7     12345.0    FALSE 5.275118     1  FALSE   FALSE
75 15     4  A3m   94.1     18148.5    FALSE 6.259632     1   TRUE   FALSE
```

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)
fitted(eUN.lmmNA, impute = TRUE, se = "total")
fitted(eUN.lmmNA, impute = TRUE, se = "total")
fitted(eUN.lmmNA, impute = TRUE, se = "total")
```

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

5.4 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[5] the results obtained with the mice package:

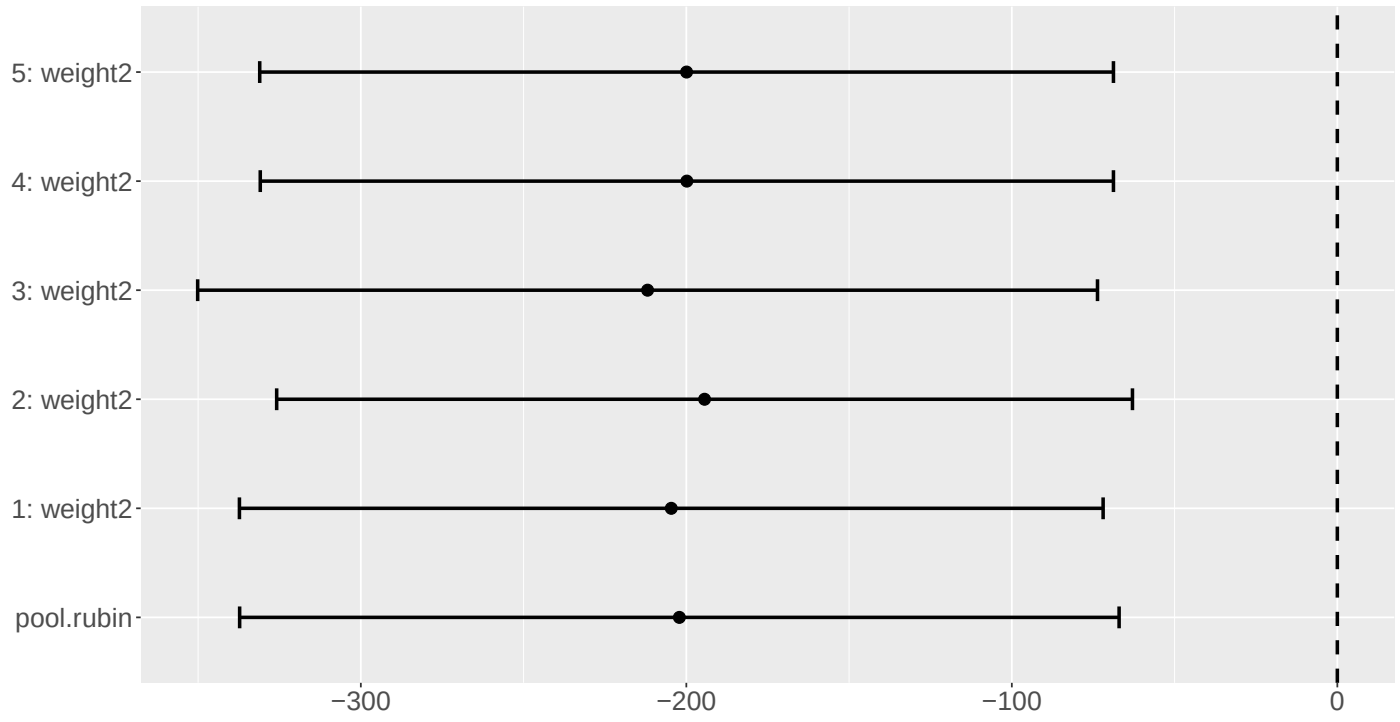
[5] almost exactly, only the degrees of freedom are a little different

```
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"))
```



6 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
```

7 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)
```

8 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] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] mice_3.14.0          lme4_1.1-29          Matrix_1.5-1         LMMstar_0.8.10
 [5] nlme_3.1-158         ggpubr_0.4.0         multcomp_1.4-20      TH.data_1.1-1
 [9] MASS_7.3-57          survival_3.3-1       mvtnorm_1.1-3        qqtest_1.2.0
[13] emmeans_1.8.1-90002 ggplot2_3.4.0

loaded via a namespace (and not attached):
 [1] tidyr_1.2.0          splines_4.2.0        carData_3.0-5        datawizard_0.6.1
 [5] assertthat_0.2.1     stats4_4.2.0         bayestestR_0.13.0    globals_0.16.1
 [9] numDeriv_2016.8-1.1 pillar_1.8.1         backports_1.4.1      lattice_0.20-45
[13] glue_1.6.2           digest_0.6.29        ggsignif_0.6.3       minqa_1.2.4
[17] colorspace_2.0-3     sandwich_3.0-2       cowplot_1.1.1        plyr_1.8.7
[21] pcaPP_2.0-2          pkgconfig_2.0.3      broom_0.8.0          listenv_0.8.0
[25] purrr_0.3.4          xtable_1.8-4         scales_1.2.1         copula_1.1-0
[29] lava_1.6.10          tibble_3.1.8         ADGofTest_0.3        mgcv_1.8-40
[33] generics_0.1.2       farver_2.1.1         car_3.1-0            withr_2.5.0
[37] cli_3.4.1            effectsize_0.7.0.5   magrittr_2.0.3       estimability_1.4.1
[41] future_1.28.0        fansi_1.0.3          parallelly_1.32.1    gsl_2.1-7.1
[45] rstatix_0.7.0        textshaping_0.3.6    tools_4.2.0          lifecycle_1.0.3
[49] pspline_1.0-19       stringr_1.4.0        munsell_0.5.0        stabledist_0.7-1
[53] compiler_4.2.0       systemfonts_1.0.4    rlang_1.0.6          grid_4.2.0
[57] nloptr_2.0.3         parameters_0.18.2    labeling_0.4.2       boot_1.3-28
[61] lmerTest_3.1-3       gtable_0.3.1         codetools_0.2-18     abind_1.4-5
[65] DBI_1.1.3            reshape2_1.4.4       R6_2.5.1             zoo_1.8-11
[69] dplyr_1.0.9          future.apply_1.9.1   utf8_1.2.2           insight_0.18.4
[73] ragg_1.2.2           stringi_1.7.6        parallel_4.2.0       Rcpp_1.0.8.3
[77] vctrs_0.5.1          tidyselect_1.1.2     coda_0.19-4
```

References

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

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. $\boldsymbol{Y}_1, \dots, \boldsymbol{Y}_n$) and $VX_1, \dots, \boldsymbol{X}_n$) along with a vector of weights $\omega = (w_1, \dots, w_n)$, 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[6]. 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`.

[6] 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[7]:

$$
\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).

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

A.4 Degrees of freedom

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

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

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

$$\mathbb{E}\left[\mathcal{H}(\theta, \theta') | \boldsymbol{X}\right] = \frac{1}{2} tr\left(A(\boldsymbol{\Theta})^{-1} \left(\sum_{i=1}^{n} w_i b_i(\boldsymbol{\Theta}) B_i(\boldsymbol{\Theta}) b_i^\intercal(\boldsymbol{\Theta}) + C(\boldsymbol{\Theta}) A(\boldsymbol{\Theta})^{-1} C^\intercal(\boldsymbol{\Theta}) \right) \right) + E(\boldsymbol{\Theta})$$

So we will first derive the derivative of $\mathcal{H}(\theta, \theta')$ 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 $\boldsymbol{X}$. To find the derivative of $\mathcal{H}(\theta, \theta')$ we can therefore take the derivative of formula (B). Its derivative with respect to the mean parameters is 0.

So we just need to compute the derivative with respect to a variance parameter θ'':

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

and

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

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), $SSE = \omega SSE^*$ where ω 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 $\varepsilon_i \sim \mathcal{N}(0, \omega\Omega)$.

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: $SSR = \omega SSR^*$.

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 Equivalent 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(coef(eCS.lmm, effects = "ranef")-ranef(eCS.lme))
```

```
[1] -3.136988e-08  2.384361e-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):

```
eCS.lmer <- lmer(weight ~ time + glucagon + (1|id),
                 data = dfL)
logLik(eCS.lmer)
logLik(eCS.lmm)
```

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

The estimated random effects match:

```
range(coef(eCS.lmm, effects = "ranef")-ranef(eCS.lmer)$id)
```

```
[1] -3.167863e-08  2.406745e-08
```

Nested random effects correspond to block unstructured:

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

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

And the estimated random effects still match:

```
eRanefBCS.lmm <- coef(eBCS.lmm, effects = "ranef")
eRanefBCS.lmer <- ranef(eBCS.lmer)
## id
range(eRanefBCS.lmm[,"id"]-eRanefBCS.lmer$id)
## baseline
range(c(eRanefBCS.lmm[,"baseline1"],eRanefBCS.lmm[,"baseline2"])-ranef(eBCS.lmer)$`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")
fm03 <- lmer(diameter ~ 1 + (1|plate) + (1|sample), Penicillin)
logLik(fm03)
```

```
'log Lik.' -165.4303 (df=4)
```

using `lmm` with a small hack: using a block compound symmetry structure with heterogeneous set to -1 to remove the correlation coefficient for pairs that have none of the covariate defining the blocks in common:

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

e.lmm <- lmm(diameter ~ 1,
             repetition = ~index|id,
             structure = CS(list(~1,~plate+sample), heterogeneous = -1),
             data = Penicillin)
logLik(e.lmm)
```

```
[1] -165.4303
```

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
)
```

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"
)
```

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

```
[1] -2.541298e-06
[1] -0.0001830095  0.0001626755
[1] -0.0003971008  0.0002047941
```

The main differences are:

- it uses the expected information matrix to quantify uncertainty instead of the observed information matrix.

- it uses the Kenward and Roger method for computing the degrees of freedom instead of the Satterthwaite approximation:
- it implements different covariance patterns
- it is faster but provides less support to work with the output of the mixed model (e.g. no `residual` method)

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
```