Overview of the package LMMstar

Brice Ozenne

June 9, 2021

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

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

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

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

```
library(LMMstar)
```

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

```
data(gastricbypassL)
head(gastricbypassL)
```

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

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

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

and rescale the glucagon values

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

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

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

```
[1] ‘0.2’
```

1 Descriptive statistics

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

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

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

2 Multivariate Gaussian model

2.1 Modeling tools

Fit a multivariate Gaussian model with **compound symmetry** structure:

```
eCS.lmm <- lmm(weight ~ time + glucagon,
        structure = CS(~time|id),
        data = gastricbypassL)
eCS.lmm
```

```
  Multivariate Gaussian Model with a compound symmetry covariance matrix

data           : 78 observations and distributed in 20 clusters
log-likelihood : -243.6005
parameters     : 5 mean ((Intercept) timeB1_week timeA1_week timeA3_months glucagon)
                 1 variance (sigma)
                 1 correlation (Rho)
```

Fit a multivariate Gaussian model with **unstructured** covariance matrix:

```
eUN.lmm <- lmm(weight ~ time + glucagon,
        structure = UN(~time|id),
        data = gastricbypassL)
eUN.lmm
```

```
  Multivariate Gaussian Model with an unstructured covariance matrix

data           : 78 observations and distributed in 20 clusters
log-likelihood : -216.3189
parameters     : 5 mean ((Intercept) timeB1_week timeA1_week timeA3_months glucagon)
                 4 variance (sigma k.B1_week k.A1_week k.A3_months)
                 6 correlation (cor(B1_week,B3_months) cor(A1_week,B3_months) cor(A3_months,B3_mont
```

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

2.2 Model output

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

```
summary(eCS.lmm, ci = TRUE)
```

```
  Multivariate Gaussian Model with a compound symmetry covariance matrix
  - fitted using Restricted Maximum Likelihood (REML)
  - log-likelihood :-243.6005 (parameters: mean = 5, variance = 1, correlation = 1)

Dataset: gastricbypassL
 - 20 clusters
 - 78 observations were analyzed, 2 were excluded because of missing values
 - 4 maximum number of observations per cluster
 - levels of the categorical variables
 - reference level: time=B3_months

$time
          B1_week A1_week A3_months
B3_months       0       0         0
B1_week         1       0         0
A1_week         0       1         0
A3_months       0       0         1

Correlation structure: ~1 | id
          B3_months B1_week A1_week A3_months
B3_months      1.00    0.97    0.97      0.97
B1_week        0.97    1.00    0.97      0.97
A1_week        0.97    0.97    1.00      0.97
A3_months      0.97    0.97    0.97      1.00

Variance structure: ~1
      standard.deviation
sigma           18.84957

Mean structure: weight ~ time + glucagon
              estimate    se     df   lower   upper p.value
(Intercept)    129.369 4.226 20.224 120.561 120.561  <0.001 ***
timeB1_week     -7.619 1.054 54.431  -9.732  -9.732  <0.001 ***
timeA1_week    -14.495 1.428  53.73 -17.358 -17.358  <0.001 ***
timeA3_months  -27.051 1.087 54.286 -29.231 -29.231  <0.001 ***
glucagon         0.822  0.62 53.053  -0.422  -0.422   0.191

The columns lower and upper correspond to the 95% confidence interval of the estimated coefficient
Note: p-values and confidence intervals are not adjusted for multiple comparisons
```

2.3 Extract estimated coefficients

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

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

```
(Intercept)   timeB1_week   timeA1_week timeA3_months      glucagon    log(sigma)     atanh(Rho)
129.3690995    -7.6194918   -14.4951323   -27.0514694     0.8217879     2.9364900      2.0911816
```

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

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

```
sigma:B3_months   sigma:B1_week   sigma:A1_week sigma:A3_months
       20.28080        19.04553        17.65479        16.76104
```

2.4 Extract estimated residual variance-covariance structure

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

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

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

It can also be specific to an individual:

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

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

2.5 Model diagnostic

The method `residuals` can be used to output the normalized residuals in a wide format:

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

This can for instance be used to check the auto-correlation between the residuals:

```
cor(eCS.diag[,-1,drop=FALSE], use = "pairwise")
```

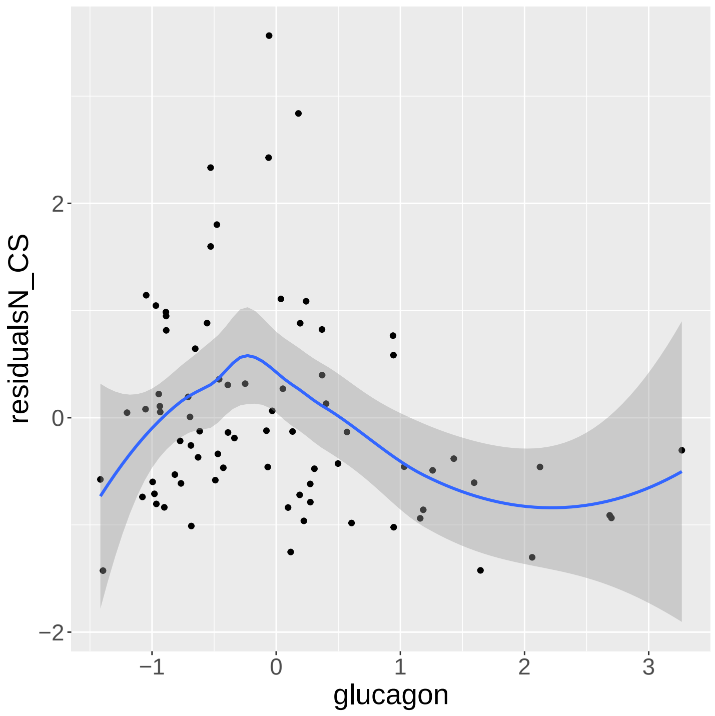
```
          B3_months   B1_week   A1_week A3_months
B3_months 1.0000000 0.6819780 0.5924644 0.3844298
B1_week   0.6819780 1.0000000 0.7996891 0.2103374
A1_week   0.5924644 0.7996891 1.0000000 0.2533221
A3_months 0.3844298 0.2103374 0.2533221 1.0000000
```

The long format:

```
gastricbypassL$residualsN_CS <- residuals(eCS.lmm, type.residual = "normalized",
       format = "long")
```

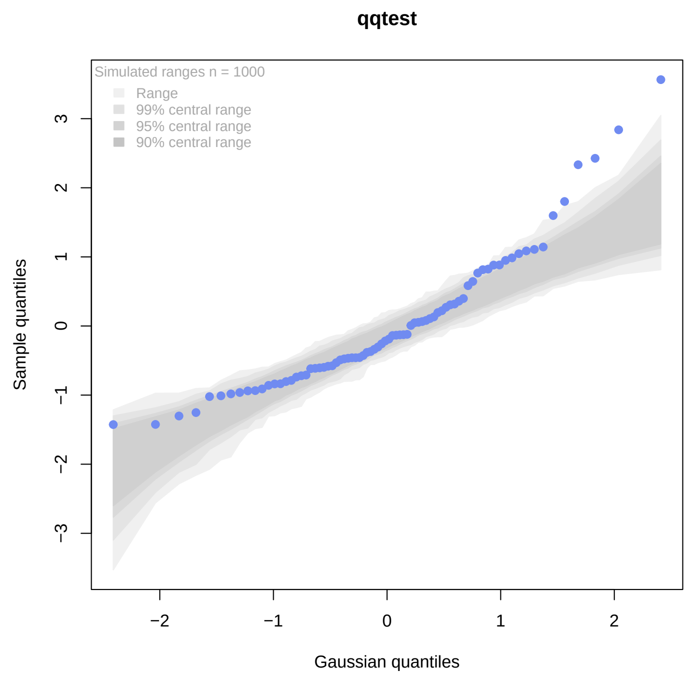
can be useful to investigate trends relative to a covariate:

```
library(ggplot2)
ggplot(gastricbypassL, aes(x=glucagon,y=residualsN_CS)) + geom_point() + geom_smooth()
```



or to look at the distribution of the residuals via a qq-plot:

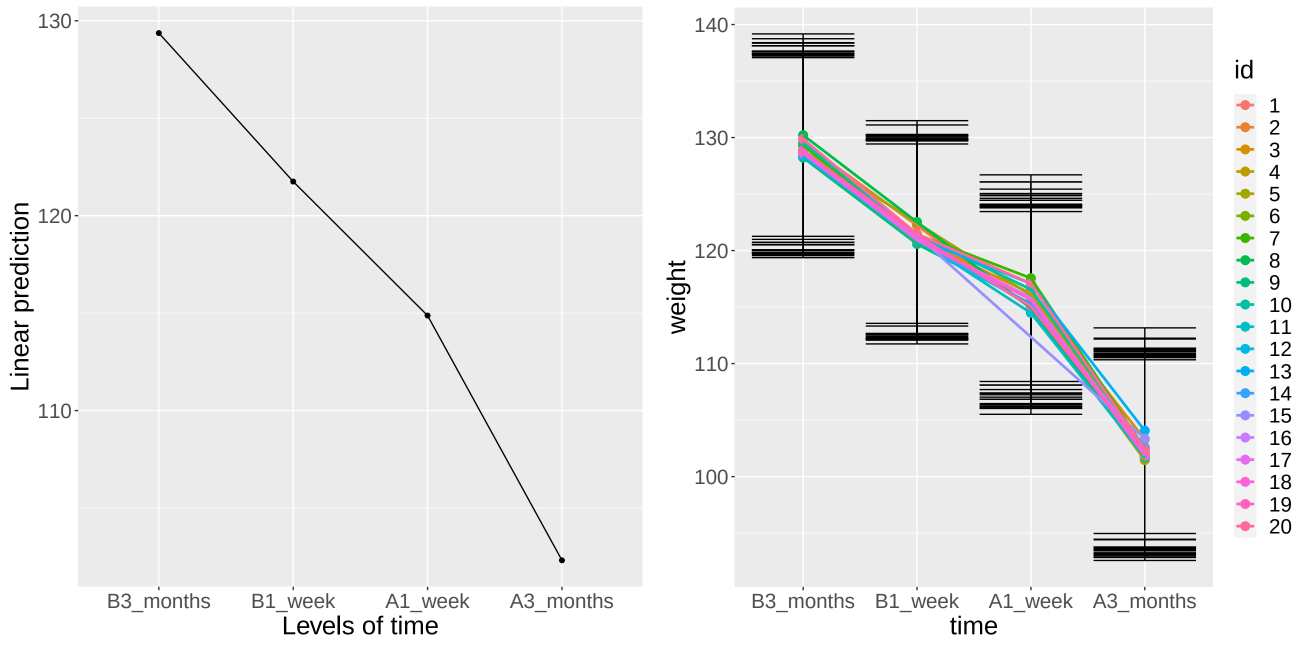
```
library(qqtest)
qqtest(na.omit(gastricbypassL$residualsN_CS))
```



2.6 Model fit

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

```
library(emmeans) ## left panel
emmip(eCS.lmm, ~time)
library(ggplot2) ## right panel
autoplot(eCS.lmm)
```



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

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

2.7 Statistical inference

2.7.1 Model coefficients

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

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

```
                 estimate        se  statistic        df       lower      upper null      p.value
(Intercept)   129.3690995 4.2256315  30.615329 20.223686 120.5608325 138.177367    0 0.000000e+00
timeB1_week    -7.6194918 1.0538287  -7.230294 54.431370  -9.7319078  -5.507076    0 1.670235e-09
timeA1_week   -14.4951323 1.4279420 -10.151066 53.729569 -17.3583136 -11.631951    0 4.263256e-14
timeA3_months -27.0514694 1.0870635 -24.884902 54.286480 -29.2306372 -24.872302    0 0.000000e+00
glucagon        0.8217879 0.6199594   1.325551 53.053075  -0.4216641   2.065240    0 1.906683e-01
log(sigma)      2.9364900 0.1580448         NA  5.518946   2.5414485   3.331532   NA           NA
atanh(Rho)      2.0911816 0.1866252  11.205249  3.251184   1.5223905   2.659973    0 1.044455e-03
```

The variance and correlation parameters being constrained parameters (e.g. strictly positive), they uncertainty is by default computed after transformation (e.g. `log`):

```
confint(eCS.lmm, effects = "variance")
```

```
           estimate        se statistic       df    lower    upper null p.value
log(sigma)  2.93649 0.1580448        NA 5.518946 2.541448 3.331532   NA      NA
```

They can be backtransformed to the original scale using `backtransform`:

```
backtransform(confint(eCS.lmm, effects = "variance"))
```

```
      estimate        se statistic       df    lower    upper null p.value
sigma 18.84957 0.1580448        NA 5.518946 12.69805 27.98116   NA      NA
Note: estimates and confidence intervals for sigma, k, rho have been back-transformed.
      standard errors are not back-transformed.
```

While not recommanded, it is also possible to not use any transformation:

```
table <- confint(eCS.lmm, effects = "variance", transform.sigma = "none")
table
```

```
      estimate       se statistic       df    lower    upper null p.value
sigma 18.84957 2.979077        NA 1.626596 2.754492 34.94464   NA      NA
```

2.7.2 Linear combination of the model coefficients

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

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

```
                     ** User-specified hypotheses **
 - F-test
 statistic df.num df.denom      p.value
  43.15392      1 17.78688 3.808793e-06

 - P-values and confidence interval (adjusted for multiplicity within each global test)
                           estimate        se       df statistic     lower     upper null
timeA1_week - timeB1_week -3.905721 0.5945537 17.78688 -6.569165 -5.155906 -2.655536    0
                               p.value
timeA1_week - timeB1_week 3.808793e-06
```

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

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

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

It is then advised to specify the null hypothesis via a contrast matrix, e.g.:

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

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

```
                    ** User-specified hypotheses **
- F-test
statistic df.num df.denom     p.value
 6.234317      3 18.02975 0.004307772
```

2.8 Baseline adjustment

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

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

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

```
gastricbypassL$treatment <- factor(gastricbypassL$group, c("none","1","2"))
gastricbypassL$treatment[gastricbypassL$time %in% c("B3_months","B1_week")] <- "none"
table(gastricbypassL$treatment, gastricbypassL$time)
```

```

      B3_months B1_week A1_week A3_months
none         20      20       0         0
1             0       0      14        14
2             0       0       6         6
```

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

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

```
 [1] "(Intercept)"              "treatment1"               "treatment2"
 [4] "timeB1_week"              "timeA1_week"              "timeA3_months"
 [7] "treatment1:timeB1_week"   "treatment2:timeB1_week"   "treatment1:timeA1_week"
[10] "treatment2:timeA1_week"   "treatment1:timeA3_months" "treatment2:timeA3_months"
```

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

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

```
Warning message:
In model.matrix_regularize(formula.mean, data) :
  Constant values in the design matrix in interactions "treatment:time"
 Coefficients "treatment1" "treatment2" "timeA1_week" "timeA3_months" "treatment1:timeB1_week" "tre
Consider defining manually the interaction, e.g. via droplevels(interaction(.,.)) to avoid this war
```

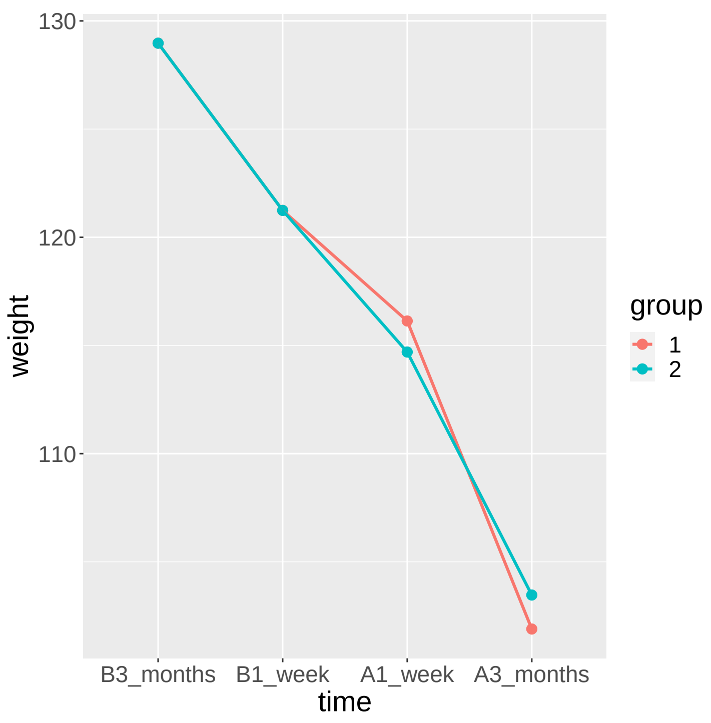
with the following coefficients:

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

```
             (Intercept)              timeB1_week   treatment1:timeA1_week   treatment2:timeA1_week
               128.97000                 -7.73000                -12.83949                -14.27452
treatment1:timeA3_months treatment2:timeA3_months
               -27.07620                -25.50553
```

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

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



3 Data generation

Simulate some data in the wide format:

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

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

Simulate some data in the long format:

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

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

4 Modifying default options

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

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

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

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

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

To restore the original default options do:

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

5 R session

Details of the R session used to generate this document:

```
sessionInfo()
```

```
R version 4.1.0 (2021-05-18)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 20.04.2 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8       
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8   
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C              
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C       

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

other attached packages:
[1] LMMstar_0.2

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.6          plyr_1.8.6          pillar_1.6.1        compiler_4.1.0     
 [5] tools_4.1.0         lifecycle_1.0.0     tibble_3.1.2        gtable_0.3.0       
 [9] nlme_3.1-152        lattice_0.20-44     pkgconfig_2.0.3     rlang_0.4.11       
[13] Matrix_1.3-3        mvtnorm_1.1-1       coda_0.19-4         stringr_1.4.0      
[17] dplyr_1.0.6         generics_0.1.0      vctrs_0.3.8         grid_4.1.0         
[21] tidyselect_1.1.1    glue_1.4.2          R6_2.5.0            fansi_0.4.2        
[25] survival_3.2-11     multcomp_1.4-17     lava_1.6.9          TH.data_1.0-10     
[29] reshape2_1.4.4      ggplot2_3.3.3       purrr_0.3.4         magrittr_2.0.1     
[33] scales_1.1.1        codetools_0.2-18    ellipsis_0.3.2      emmeans_1.6.0      
[37] MASS_7.3-54         splines_4.1.0       xtable_1.8-4        colorspace_2.0-1   
[41] numDeriv_2016.8-1.1 sandwich_3.0-1      utf8_1.2.1          stringi_1.6.2      
[45] estimability_1.3    munsell_0.5.0       crayon_1.4.1        zoo_1.8-9          
```

References