REML vs ML
with an aside on cvar and dvar

MATH 4939

2025-03-30

```
#
# Normally I would hide the following
# two lines of code by heading
# the chunk with '#+ include=FALSE' but I'm
# including them to show how you can run a
# script in which you don't want errors to
# to stop execution but you want the error
```

```r
# output to be shown, and how to control the
# width of R output.
#
knitr::opts_chunk$set(error = TRUE, comment = '    ')
options(width=60)
```

- ML: Maximum Likelihood
  - using Y as data
  - maximized over all parameters, FE and RE
- REML: Residual (or Restricted) Maximum Likehood
  - using residuals Y from X as data
  - maximized over RE parameters, i.e. parameters in the G and R matrices

Recall:

- You can compare two models using ML **only if they have exactly the same response values** but possibly different parameters.
- If one model is nested in the other, you can use a Likelihood Ratio Test (LRT or Wilk's test) with the smaller model as a **null** hypothesis and the larger model as an **alternative** hypothesis. Asymptotically:

2 × difference in maximum log-likelihoods

has a Chi-squared distribution with dfs equal to the difference in degrees of freedom, if the null hypothesis is correct.
  - In R, you can often use:
    `anova(fit0, fit1)`
- If the models are not nested you might consider comparing them with AIC or BIC, etc.
  - In R, you can often use: `AIC(fit0, fit1)` or `BIC(fit0, fit1)`
- **Beware:** If you have missing data, larger models will often drop observations that are not dropped in a smaller model. To use an LRT you need to refit the smaller model using only the data used in the larger model.
- Another way to test a hypothesis is to use a **Wald test**. When you use a Wald test you have one model that plays the role of the **alternative** model, and the **null** model is included by in the alternative model by specifying that some parameters are fixed, typically to be equal to 0.
  - In R, you can use: `car::lht` or `spida2::wald`.
  - In this case, the 'two' models are automatically fitted on the same data and the ___null_ hypothesis is automatically nested in the **alternative**.
  - Downside: Wald test only work easily for linear hypotheses involving FE

parameters.

In 'nlme', models use **REML** by default. You can require **ML** with the argument: `method = ML`.

It is thought that **REML** gives better estimates of the `G` and `R` matrices and more accurate Wald tests.

Downside to **REML**:

- You can use LRT (`anova`) to compare two **REML** fits **only if they have equivalent FE models**. Why? Because the data are the **residuals from the X matrix** and if the FE models are not equivalent, then the residuals are different.

Rule 1: You can use LRT tests to compare two **REML** fits **only if they have equivalent FE models**. That is, you can use LRT tests to compare **REML** models **only** to compare the RE portion of the models.

How do you compare two FE models with the same or different RE models? **You must refit the two models using ML**.

So if `fit1` and `fit2` were fitted using **REML** (the usual default), you can do this:

```
anova(update(fit1, method = 'ML'), update(fit2, method = 'ML'))
```

Of course, one of the models, `fit1` or `fit2`, must be nested in the other for the result to be interpreted as a p-value for a hypothesis test.

Some examples:

```
library(nlme)
library(car)
```

```
    Loading required package: carData
```

```
library(lattice)
library(spida2)
```

```
    Attaching package: 'spida2'

    The following object is masked from 'package:nlme':

        getData
```

```
library(latticeExtra)
Layer <- latticeExtra::layer   # to circumvent problems with the tidyverse
```

```r
# previous fit with REML (default)

head(hs)
```

```
  school mathach    ses    Sex Minority Size   Sector
1   1317  12.862  0.882 Female       No  455 Catholic
2   1317   8.961  0.932 Female      Yes  455 Catholic
3   1317   4.756 -0.158 Female      Yes  455 Catholic
4   1317  21.405  0.362 Female      Yes  455 Catholic
5   1317  20.748  1.372 Female       No  455 Catholic
6   1317  18.362  0.132 Female      Yes  455 Catholic
  PRACAD DISCLIM
1   0.95  -1.694
2   0.95  -1.694
3   0.95  -1.694
4   0.95  -1.694
5   0.95  -1.694
6   0.95  -1.694
```

```
hs$id <- as.factor(hs$school)
dd <- hs

fit.reml <- lme( mathach ~ ses * Sector, dd, random = ~ 1 + ses | id,
            control = list(msMaxIter=200, msVerbose=T))
```

```
      0:     11105.191:  1.22722  2.27855 -3.36583
      1:     11105.171:  1.23607  3.08949 -3.39035
      2:     11105.163:  1.21688  3.08502 -3.39105
      3:     11105.132:  1.21784  2.92394 -3.40142
      4:     11105.097:  1.23154  2.70125 -3.42484
      5:     11105.089:  1.21583  2.62466 -3.43752
      6:     11105.087:  1.22470  2.55186 -3.46743
      7:     11105.084:  1.22539  2.58263 -3.50346
      8:     11105.081:  1.22624  2.60459 -3.59832
      9:     11105.070:  1.23049  2.65105 -3.98501
     10:     11105.047:  1.24358  2.72641 -5.13118
     11:     11105.030:  1.25874  2.75845 -6.27935
     12:     11105.016:  1.28761  2.77512 -7.96707
     13:     11105.012:  1.31100  2.78975 -8.94137
```

```
     14:      11105.011:  1.32455  2.80259 -9.59086
     15:      11105.011:  1.33280  2.81718 -10.0206
     16:      11105.011:  1.33783  2.82963 -10.3563
     17:      11105.011:  1.34009  2.83987 -10.5400
     18:      11105.011:  1.33984  2.84316 -10.5748
     19:      11105.011:  1.33932  2.84331 -10.5539
```

```
getG(fit.reml)
```

```
    Random effects variance covariance matrix
                (Intercept)     ses
    (Intercept)      3.5405 0.35100
    ses              0.3510 0.12693
      Standard Deviations: 1.8816 0.35627
```

```
svd(getG(fit.reml))$d
```

```
    [1] 3.57619168 0.09120944
```

fit with method = "ML"

```
fit.ml <- lme( mathach ~ ses * Sector, dd, random = ~ 1 + ses | id,
               na.action = na.exclude, method = "ML",
```

```
              control = list(msMaxIter=200, msVerbose=T, returnObject = T))
```

```
 0:     11116.488:  1.26387  2.36435 -3.50300
 1:     11116.369:  1.27189  3.26343 -3.52690
 2:     11116.362:  1.25371  3.26152 -3.52738
 3:     11116.357:  1.26159  3.19060 -3.54350
 4:     11116.344:  1.26820  3.00295 -3.58152
 5:     11116.336:  1.27032  2.81689 -3.62715
 6:     11116.332:  1.26359  2.81940 -3.65385
 7:     11116.330:  1.25804  2.81818 -3.71168
 8:     11116.329:  1.25586  2.81903 -3.76974
 9:     11116.324:  1.25192  2.81641 -4.00212
10:     11116.314:  1.24864  2.82898 -4.61256
11:     11116.304:  1.25175  2.85728 -5.22247
12:     11116.277:  1.27266  2.93766 -7.00675
13:     11116.253:  1.30768  3.05742 -9.17625
14:     11116.240:  1.34230  3.01265 -11.3486
15:     11116.229:  1.37864  3.10910 -13.5192
16:     11116.224:  1.41459  3.14638 -15.6917
17:     11116.221:  1.45115  3.15181 -17.8645
```

```
18:     11116.220:  1.48587  3.18162 -20.0371
19:     11116.219:  1.51997  3.20433 -22.2098
20:     11116.218:  1.55045  3.22385 -24.2232
21:     11116.218:  1.57933  3.24623 -26.2367
22:     11116.217:  1.60668  3.26415 -28.2502
23:     11116.217:  1.63083  3.28580 -30.0664
24:     11116.217:  1.65207  3.30801 -31.8826
25:     11116.216:  1.67435  3.34695 -34.5049
26:     11116.215:  1.69681  3.38707 -37.1271
27:     11116.215:  1.68879  3.41381 -37.9533
28:     11116.215:  1.69560  3.41771 -38.3666
29:     11116.215:  1.70007  3.41704 -38.5714
30:     11116.215:  1.70369  3.41672 -38.7763
31:     11116.215:  1.71560  3.41893 -39.5960
32:     11116.215:  1.74187  3.42461 -41.4810
33:     11116.215:  1.75375  3.42421 -42.2823
34:     11116.215:  1.78632  3.44180 -45.1199
35:     11116.214:  1.79076  3.44730 -45.7488
36:     11116.214:  1.81616  3.46496 -48.2647
37:     11116.214:  1.82977  3.47445 -49.7050
```

```
38:     11116.214:  1.84527  3.48110 -51.1453
39:     11116.214:  1.87702  3.49064 -54.0324
40:     11116.214:  1.90875  3.50216 -56.9194
41:     11116.214:  1.89671  3.49707 -55.8261
42:     11116.214:  1.90804  3.50081 -56.8877
43:     11116.214:  1.92497  3.50653 -58.4927
44:     11116.214:  1.94170  3.51195 -60.0977
45:     11116.214:  1.95335  3.51533 -61.2471
46:     11116.214:  1.97183  3.52112 -63.1088
47:     11116.214:  1.99514  3.52846 -65.4758
48:     11116.214:  2.01737  3.53505 -67.8429
49:     11116.214:  2.04035  3.54179 -70.2100
50:     11116.214:  2.03515  3.53965 -69.6645
51:     11116.214:  2.04048  3.54080 -70.2100
52:     11116.214:  2.06164  3.54565 -72.3922
53:     11116.214:  2.07673  3.54865 -73.9579
54:     11116.214:  2.09160  3.55088 -75.5237
55:     11116.214:  2.10631  3.55397 -77.0894
56:     11116.214:  2.12045  3.55731 -78.6552
57:     11116.214:  2.12906  3.56011 -79.7143
```

```
 58:     11116.214:  2.14815  3.56499 -81.9190
 59:     11116.214:  2.15654  3.56716 -82.9539
 60:     11116.214:  2.16555  3.56907 -83.9888
 61:     11116.214:  2.17468  3.57068 -85.0237
 62:     11116.214:  2.18405  3.57184 -86.0586
 63:     11116.214:  2.20071  3.57288 -87.8513
 64:     11116.214:  2.20966  3.57317 -88.8229
 65:     11116.214:  2.21823  3.57423 -89.7944
 66:     11116.214:  2.22652  3.57551 -90.7660
 67:     11116.214:  2.23440  3.57712 -91.7376
 68:     11116.214:  2.24799  3.58088 -93.5302
 69:     11116.214:  2.25427  3.58296 -94.4213
 70:     11116.214:  2.26132  3.58441 -95.3125
 71:     11116.214:  2.26849  3.58562 -96.2036
 72:     11116.214:  2.27588  3.58647 -97.0947
 73:     11116.214:  2.29042  3.58730 -98.7947
 74:     11116.214:  2.29801  3.58690 -99.6379
 75:     11116.214:  2.30492  3.58759 -100.481

Warning in lme.formula(mathach ~ ses * Sector, dd, random = ~1 + ses | id,
  message = singular convergence (7)
```

did not converge

```
getG(fit.ml)
```

```
    Random effects variance covariance matrix
                (Intercept)      ses
    (Intercept)     3.25250 0.287270
    ses             0.28727 0.028656
      Standard Deviations: 1.8035 0.16928
```

```
svd(getG(fit.ml))$d
```

```
    [1] 3.27788429 0.00325807
```

equivalent:

```
fit.ml <- update( fit, method = 'ML')
```

```
    Error in eval(expr, envir, enclos): object 'fit' not found
```

Note takes more iterations but gives up with 'singular convergence' probably because estimate RE variance is 'smaller' and closer to singular

Note: increasing msMaxIter won't help

- One solution is to recenter ses in the RE model
- Another is to use dvar in the RE model

```
clist <- list(msMaxIter = 200, msVerbose = T, returnObject = TRUE)
```

The magic of a more sensible model:

```
fit2.reml <- lme(mathach ~ (ses + cvar(ses, id)) * Sector, dd,
                 random = ~ 1 + ses | id,
                 control = clist)
```

```
      0:     11089.451:  1.43803  2.20283  1.16160
      1:     11089.398:  1.45004  2.50336  1.16921
      2:     11089.395:  1.43660  2.49886  1.17032
      3:     11089.392:  1.44694  2.46849  1.17721
      4:     11089.388:  1.43988  2.40664  1.19799
      5:     11089.387:  1.43866  2.41179  1.22947
      6:     11089.379:  1.43555  2.43748  1.61468
      7:     11089.373:  1.44005  2.45292  2.00041
      8:     11089.369:  1.45155  2.46709  2.45283
      9:     11089.368:  1.45763  2.46890  2.60772
     10:     11089.368:  1.45894  2.46706  2.63093
```

```
     11:     11089.368:  1.45904  2.46653  2.62577
```

```
getG(fit2.reml)
```

```
    Random effects variance covariance matrix
                (Intercept)      ses
    (Intercept)     2.12030 -0.16438
    ses            -0.16438  0.26931
      Standard Deviations: 1.4561 0.51895
```

A different RE model

```
fit3.reml <- lme(mathach ~ (ses + cvar(ses, id)) * Sector, dd,
                 random = ~ 1 + dvar(ses, id) | id,
                 control = clist)
```

```
      0:     11089.213:  1.42981  2.05867 0.379581
      1:     11089.186:  1.43155  2.25208 0.382026
      2:     11089.184:  1.42828  2.23384 0.383640
      3:     11089.184:  1.43930  2.21894 0.385312
      4:     11089.182:  1.43028  2.21668 0.385731
      5:     11089.181:  1.43335  2.19861 0.388925
```

```
      6:     11089.181:  1.42941  2.18694 0.401569
      7:     11089.181:  1.43023  2.18886 0.414258
      8:     11089.180:  1.43305  2.19585 0.523219
      9:     11089.179:  1.43319  2.19701 0.607454
     10:     11089.179:  1.43241  2.19579 0.642267
     11:     11089.179:  1.43211  2.19517 0.641239
     12:     11089.179:  1.43206  2.19503 0.640007
```

```
getG(fit3.reml)
```

```
    Random effects variance covariance matrix
                 (Intercept) dvar(ses, id)
    (Intercept)      2.138900     -0.070718
    dvar(ses, id)   -0.070718      0.462680
      Standard Deviations: 1.4625 0.68021
```

```
AIC(fit2.reml, fit3.reml)  # ok because both have the same FE
```

```
              df      AIC
    fit2.reml 10 12839.60
    fit3.reml 10 12839.22
```

What happens if we use an LRT to compare fit.reml with fit2.reml? That would be a test of dropping cvar(ses, id)

```
anova(fit.reml, fit2.reml)
```

```
    Warning in anova.lme(fit.reml, fit2.reml): fitted objects
    with different fixed effects. REML comparisons are not
    meaningful.

              Model df      AIC      BIC    logLik   Test
    fit.reml      1  8 12855.39 12900.09 -6419.695
    fit2.reml     2 10 12839.60 12895.47 -6409.801 1 vs 2
               L.Ratio p-value
    fit.reml
    fit2.reml 19.78691   1e-04
```

```
anova(update(fit.reml, method = 'ML'), update(fit2.reml, method = 'ML'))
```

```
      0:     11116.488:  1.26387  2.36435 -3.50300
      1:     11116.369:  1.27189  3.26343 -3.52690
      2:     11116.362:  1.25371  3.26152 -3.52738
      3:     11116.357:  1.26159  3.19060 -3.54350
```

```
 4:     11116.344:  1.26820  3.00295 -3.58152
 5:     11116.336:  1.27032  2.81689 -3.62715
 6:     11116.332:  1.26359  2.81940 -3.65385
 7:     11116.330:  1.25804  2.81818 -3.71168
 8:     11116.329:  1.25586  2.81903 -3.76974
 9:     11116.324:  1.25192  2.81641 -4.00212
10:     11116.314:  1.24864  2.82898 -4.61256
11:     11116.304:  1.25175  2.85728 -5.22247
12:     11116.277:  1.27266  2.93766 -7.00675
13:     11116.253:  1.30768  3.05742 -9.17625
14:     11116.240:  1.34230  3.01265 -11.3486
15:     11116.229:  1.37864  3.10910 -13.5192
16:     11116.224:  1.41459  3.14638 -15.6917
17:     11116.221:  1.45115  3.15181 -17.8645
18:     11116.220:  1.48587  3.18162 -20.0371
19:     11116.219:  1.51997  3.20433 -22.2098
20:     11116.218:  1.55045  3.22385 -24.2232
21:     11116.218:  1.57933  3.24623 -26.2367
22:     11116.217:  1.60668  3.26415 -28.2502
23:     11116.217:  1.63083  3.28580 -30.0664
```

```
24:     11116.217:  1.65207  3.30801 -31.8826
25:     11116.216:  1.67435  3.34695 -34.5049
26:     11116.215:  1.69681  3.38707 -37.1271
27:     11116.215:  1.68879  3.41381 -37.9533
28:     11116.215:  1.69560  3.41771 -38.3666
29:     11116.215:  1.70007  3.41704 -38.5714
30:     11116.215:  1.70369  3.41672 -38.7763
31:     11116.215:  1.71560  3.41893 -39.5960
32:     11116.215:  1.74187  3.42461 -41.4810
33:     11116.215:  1.75375  3.42421 -42.2823
34:     11116.215:  1.78632  3.44180 -45.1199
35:     11116.214:  1.79076  3.44730 -45.7488
36:     11116.214:  1.81616  3.46496 -48.2647
37:     11116.214:  1.82977  3.47445 -49.7050
38:     11116.214:  1.84527  3.48110 -51.1453
39:     11116.214:  1.87702  3.49064 -54.0324
40:     11116.214:  1.90875  3.50216 -56.9194
41:     11116.214:  1.89671  3.49707 -55.8261
42:     11116.214:  1.90804  3.50081 -56.8877
43:     11116.214:  1.92497  3.50653 -58.4927
```

```
44:     11116.214:  1.94170  3.51195 -60.0977
45:     11116.214:  1.95335  3.51533 -61.2471
46:     11116.214:  1.97183  3.52112 -63.1088
47:     11116.214:  1.99514  3.52846 -65.4758
48:     11116.214:  2.01737  3.53505 -67.8429
49:     11116.214:  2.04035  3.54179 -70.2100
50:     11116.214:  2.03515  3.53965 -69.6645
51:     11116.214:  2.04048  3.54080 -70.2100
52:     11116.214:  2.06164  3.54565 -72.3922
53:     11116.214:  2.07673  3.54865 -73.9579
54:     11116.214:  2.09160  3.55088 -75.5237
55:     11116.214:  2.10631  3.55397 -77.0894
56:     11116.214:  2.12045  3.55731 -78.6552
57:     11116.214:  2.12906  3.56011 -79.7143
58:     11116.214:  2.14815  3.56499 -81.9190
59:     11116.214:  2.15654  3.56716 -82.9539
60:     11116.214:  2.16555  3.56907 -83.9888
61:     11116.214:  2.17468  3.57068 -85.0237
62:     11116.214:  2.18405  3.57184 -86.0586
63:     11116.214:  2.20071  3.57288 -87.8513
```

```
 64:     11116.214:  2.20966  3.57317 -88.8229
 65:     11116.214:  2.21823  3.57423 -89.7944
 66:     11116.214:  2.22652  3.57551 -90.7660
 67:     11116.214:  2.23440  3.57712 -91.7376
 68:     11116.214:  2.24799  3.58088 -93.5302
 69:     11116.214:  2.25427  3.58296 -94.4213
 70:     11116.214:  2.26132  3.58441 -95.3125
 71:     11116.214:  2.26849  3.58562 -96.2036
 72:     11116.214:  2.27588  3.58647 -97.0947
 73:     11116.214:  2.29042  3.58730 -98.7947
 74:     11116.214:  2.29801  3.58690 -99.6379
 75:     11116.214:  2.30492  3.58759 -100.481
```

```
Error in lme.formula(fixed = mathach ~ ses * Sector, data = dd, random = ~1
  message = singular convergence (7)
```

OOps – won't work

But we can use a Wald test on the larger model

```
fit2.reml
```

```
Linear mixed-effects model fit by REML
```

```
  Data: dd
  Log-restricted-likelihood: -6409.801
  Fixed: mathach ~ (ses + cvar(ses, id)) * Sector
               (Intercept)                         ses
                13.3560596                   1.6994006
             cvar(ses, id)                SectorPublic
                 2.4078087                  -0.9343383
          ses:SectorPublic cvar(ses, id):SectorPublic
                 1.1730918                   1.4148474

Random effects:
 Formula: ~1 + ses | id
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev   Corr
(Intercept) 1.456131 (Intr)
ses         0.518950 -0.218
Residual    6.114006

Number of Observations: 1977
Number of Groups: 40
```

```
wald(fit2.reml, 'cvar')
```

```
$cvar
    numDF denDF  F-value p-value
cvar     2    36 9.749805 0.00041

$`cvar(ses, id)`
                            Estimate Std.Error DF t-value
cvar(ses, id)               2.407809 1.062916  36 2.265285
cvar(ses, id):SectorPublic  1.414847 1.465201  36 0.965633
                            p-value Lower 0.95 Upper 0.95
cvar(ses, id)               0.02961  0.252115  4.563503
cvar(ses, id):SectorPublic  0.34067 -1.556719  4.386414
```

We can use anova (LRT) to compare nested RE models with the same FE models

```
fit3.reml <- lme(mathach ~ (ses + cvar(ses, id)) * Sector, dd,
                 random = ~ 1 + dvar(ses, id) | id,
                 control = clist)
```

```
      0:     11089.213:  1.42981  2.05867 0.379581
      1:     11089.186:  1.43155  2.25208 0.382026
      2:     11089.184:  1.42828  2.23384 0.383640
      3:     11089.184:  1.43930  2.21894 0.385312
```

```
      4:     11089.182:  1.43028  2.21668 0.385731
      5:     11089.181:  1.43335  2.19861 0.388925
      6:     11089.181:  1.42941  2.18694 0.401569
      7:     11089.181:  1.43023  2.18886 0.414258
      8:     11089.180:  1.43305  2.19585 0.523219
      9:     11089.179:  1.43319  2.19701 0.607454
     10:     11089.179:  1.43241  2.19579 0.642267
     11:     11089.179:  1.43211  2.19517 0.641239
     12:     11089.179:  1.43206  2.19503 0.640007
```

```
fit4.reml <- lme(mathach ~ (ses + cvar(ses, id)) * Sector, dd,
                 random = ~ 1  | id,
                 control = clist)
```

```
      0:     11089.712:  1.43298
      1:     11089.712:  1.43298
```

```
anova(fit3.reml, fit4.reml)  # same FE models, nested RE models
```

```
             Model df      AIC      BIC    logLik   Test
fit3.reml        1 10 12839.22 12895.09 -6409.612
fit4.reml        2  8 12836.29 12880.98 -6410.145 1 vs 2
```

```
            L.Ratio p-value
fit3.reml
fit4.reml 1.066192  0.5868
```

There is no evidence that we need a random slope.

However, if we want to test the FE model for slopes and generalize it to the population of schools, the conclusions are more convincing if the model includes random school-to-school variability in slopes.

Summary

The default for lme is fitting with 'Residual Maximum Likelihood' (REML) – it has other interpretations but everyone agrees on 'REML' – With REML, maximum likelihood is applied ONLY to the parameters for random effects, i.e. the G matrix and sigma (or R). The Gammas play no direct role, they are only estimated with Generalized Least Squares as an afterthought. This means that the likelihood with REML is only informative for the the RE model: the G and R matrices. You can't compare two REML models that have different FE models.

Full maximum likelihood (ML) does look at Gammas, G and R together. So the likelihood with ML can be used to test two models with different Gammas

The consequence is that:

If two models have been fitted with REML, the must have identical FE models to compare them with 'anova': p-values, AIC, BIC.

If two models have been fitted with ML, you can use 'anova' regardless of the FE or the RE models — PROVIDED they have the same response variable.

p-values are meaningful for tests only if one model is a submodel of the other.

Wald tests can be used for both ML or REML fits. They might be more accurate with REML fits. However, Wald test are useful mainly for the FE part of the model. 'wald' in 'spida2' only works for the FE model. To compare two RE models (with identical FE models) LRT is generally better than Wald. However, classic p-values tend to be too large in many cases and should be adjusted through simulation.

Summary: - wald is ok for the FE model whether ML or REML - anova (with simulation adjustment) is ok for the RE model provided the FE models are identical - anova can be used to test different FE models or models that differ in both FEs and REs ONLY IF THE MODELS ARE fitted with ML.