Identifiability of the Random Effects Model

Is Your Model Too Big for Your Data?

georges

2022-04-04

```
library(nlme)
library(spida2)
```

```
## 
## Attaching package: 'spida2'

## The following object is masked from 'package:nlme':
## 
##     getData
```

```
set.seed(123)
dd <- expand.grid(id = 1:10000, time = c(-1,1))
#
# setting parameters
#
gamma = c(1,2)
G <- cbind(c(2,.5),c(.5,1)) # check: is this a variance matrix?
sigma <- 0.1
#
# Note:
#
Z <- cbind(1, c(-1,1))
Z
```

```
     [,1] [,2]
[1,]    1   -1
[2,]    1    1
```

```
V <- Z %*% G %*% t(Z) + sigma^2 * diag(2)
V
```

```
     [,1] [,2]
[1,] 2.01 1.00
[2,] 1.00 4.01
```

```
#
# To generate Us we need a function to factor G
# but you could use the package 'mvtnorm' to generate
# multivariate normal observations.
#
rightfactor <- function(G) {
  # Warning: this only works correctly if G is non-negative definite
  fac <- svd(G, nu = 0) # G = UDV' with D nnd diagonal and V orthogonal
  sqrt(fac$d) * t(fac$v)
```

```r
}
#
# Check:
#
crossprod(rightfactor(G))
```

```
         [,1] [,2]
    [1,]  2.0  0.5
    [2,]  0.5  1.0
```

```r
#
# Generating u's and epsilons
#
K <- length(unique(dd$id))
Us <- matrix(rnorm(K*2), K, 2) %*% rightfactor(G)
Eps <- sigma * rnorm(K*2)
#
# Out of curiosity:
#
var(Us)
```

```
              [,1]      [,2]
    [1,] 2.0008649 0.4913603
    [2,] 0.4913603 0.9956085
```

```r
var(Eps)
```

```
    [1] 0.009995881
```

```r
#
# Finishing our data frame:
#
dd <- within(
  dd,
  {
    y <- cbind(1, time) %*% gamma +
      rowSums(cbind(1, time) * Us[id,]) +  # check that this works
      Eps
  }
)
#
# Fitting a model:
#
fit <- lme(y ~ time, dd, random = ~ 1 + time | id)
summary(fit)
```

```
    Linear mixed-effects model fit by REML
      Data: dd
           AIC      BIC    logLik
      76339.44 76386.86 -38163.72

    Random effects:
     Formula: ~1 + time | id
     Structure: General positive-definite, Log-Cholesky parametrization
                StdDev    Corr
    (Intercept) 1.3536086 (Intr)
```

```
time        0.9104714 0.4
Residual    0.5892146

Fixed effects:  y ~ time
               Value  Std.Error   DF   t-value p-value
(Intercept) 1.005741 0.01416278 9999  71.01298       0
time        1.993950 0.01001272 9999 199.14170       0
 Correlation:
     (Intr)
time 0.348

Standardized Within-Group Residuals:
          Min            Q1           Med            Q3           Max
-1.8066184703 -0.2508345721 -0.0000369109  0.2498123548  1.4912448416

Number of Observations: 20000
Number of Groups: 10000
```

```
intervals(fit)
```

```
Approximate 95% confidence intervals

 Fixed effects:
                lower     est.    upper
(Intercept) 0.9779793 1.005741 1.033503
time        1.9743226 1.993950 2.013576

 Random Effects:
  Level: id
                          lower      est.     upper
sd((Intercept))       1.3325105 1.3536086 1.3750408
sd(time)              0.8980416 0.9104714 0.9230733
cor((Intercept),time) 0.3762552 0.4004818 0.4241611

 Within-group standard error:
    lower      est.     upper
0.5477823 0.5892146 0.6337808
```

```
#
# Compare these confidence intervals with the true values
#
sqrt(diag(G))
```

```
[1] 1.414214 1.000000
```

```
cov2cor(G)
```

```
          [,1]      [,2]
[1,] 1.0000000 0.3535534
[2,] 0.3535534 1.0000000
```

```
sigma
```

```
[1] 0.1
```

```
#
# How close is V?
#
```

```r
getV(fit)
```

```
id 1
Marginal variance covariance matrix
       1      2
1 2.0213 1.0033
2 1.0033 3.9955
  Standard Deviations: 1.4217 1.9989
```

```r
V
```

```
     [,1] [,2]
[1,] 2.01 1.00
[2,] 1.00 4.01
```

```r
getV(fit)[[1]] - V
```

```
           1           2
1 0.01126330  0.00329813
2 0.00329813 -0.01448665
```

```r
#
# How close is R?
#
getR(fit)
```

```
id 1
Conditional variance covariance matrix
        1       2
1 0.34717 0.00000
2 0.00000 0.34717
  Standard Deviations: 0.58921 0.58921
```

```r
getR(fit)[[1]] - sigma^2 * diag(2)
```

```
          1         2
1 0.3371739 0.0000000
2 0.0000000 0.3371739
```

```r
#
# How close is G?
#
getG(fit)
```

```
Random effects variance covariance matrix
            (Intercept)    time
(Intercept)     1.83230 0.49356
time            0.49356 0.82896
  Standard Deviations: 1.3536 0.91047
```

```r
unclass(getG(fit)) - G
```

```
             (Intercept)         time
(Intercept) -0.167743717 -0.006437488
time        -0.006437488 -0.171041847
attr(,"group.levels")
[1] "id"
```

```r
#
# Moral:
#
# You need to check the random part of the model for identifiability.
#
# Exercise:
#
# - Rerun the example using different random seeds. You will find a number
#   of different results with the same parameters and model:
#   - non-convergence
#   - convergence but intervals(fit) will give an error because the
#     Hessian matrix is singular
#   - convergence to results that don't give correct estimates of G and R
# - Generate the example above but with 3 time points, -1, 0 and 1.
# - Try the following model on
#
# Here's another model that shouldn't work, but does:
fit <- lme(y ~ 1 + time, dd, random = ~ 1 | id, corr = corAR1(form = ~ 1|id))
# Note
summary(fit)
```

```
Linear mixed-effects model fit by REML
  Data: dd
      AIC      BIC   logLik
  77628.6 77668.11 -38809.3

Random effects:
 Formula: ~1 | id
        (Intercept) Residual
StdDev:    1.000809 1.416605

Correlation Structure: AR(1)
 Formula: ~1 | id
 Parameter estimate(s):
         Phi
0.0008364802
Fixed effects:  y ~ 1 + time
               Value  Std.Error   DF   t-value p-value
(Intercept) 1.005741 0.01416278 9999  71.01299       0
time        1.993950 0.01001272 9999 199.14167       0
 Correlation:
     (Intr)
time 0

Standardized Within-Group Residuals:
         Min           Q1          Med           Q3          Max
-3.674348390 -0.567710899  0.001814345  0.561140226  3.532166792

Number of Observations: 20000
Number of Groups: 10000
```

```r
#
# Good exam question: analyze the model above for identifiability
#
```

```r
# Here's a model that works but doesn't capture the randomness in the
# generating process:
#
fit <- lme(y ~ 1 + time, dd, random = ~ 1 | id)
summary(fit)
```

```
Linear mixed-effects model fit by REML
  Data: dd
      AIC      BIC   logLik
  77626.6 77658.21 -38809.3

Random effects:
 Formula: ~1 | id
        (Intercept) Residual
StdDev:    1.001648 1.416012

Fixed effects:  y ~ 1 + time
               Value  Std.Error   DF   t-value p-value
(Intercept) 1.005741 0.01416278 9999  71.01298       0
time        1.993950 0.01001272 9999 199.14170       0
 Correlation:
     (Intr)
time 0

Standardized Within-Group Residuals:
         Min           Q1          Med           Q3          Max
-3.673604292 -0.567692590  0.001822975  0.560536108  3.531588352

Number of Observations: 20000
Number of Groups: 10000
```

```r
intervals(fit)
```

```
Error in intervals.lme(fit): cannot get confidence intervals on var-cov components: Non-positive de
 Consider 'which = "fixed"'
```

```r
#
# Note that V is constrained to be diagonal and isn't fitting the true V
#
getV(fit)
```

```
id 1
Marginal variance covariance matrix
       1      2
1 3.0084 1.0033
2 1.0033 3.0084
  Standard Deviations: 1.7345 1.7345
```

A solution for 'shortitudinal' data

If you have data that is measured at relatively integer valued time points you can consider using 'gls' to generate identifiably a correct V matrix.

'gls' only allows:

- correlation argument:

- e.g. `corr = corAR1(form = ~time | id)` where 'time' must be integer valued
- `corr = corSymm(form = ~time |id)` to get the same correlation everywhere. This produces almost the same V matrix as `lme ... random = ~ 1 |id`
- 'time' should take on consecutive integers. Some clusters can miss some times but no time should be missing in all clusters.

- weights argument allowing heteroskedasticity
  - `weights = varIdent(form = ~ 1 | time)` will allow different variances at different times.

Combining correlation and weights allows you to fit an identifiably parametrized V matrix.

```
#
head(dd)
```

```
  id time          y
1  1   -1 -3.3911532
2  2   -1 -0.6429173
3  3   -1 -3.5420310
4  4   -1 -0.5624700
5  5   -1 -1.4756211
6  6   -1 -3.8624856
```

```
tab(dd, ~ time)
```

```
time
   -1     1 Total
10000 10000 20000
```

```
dd$occ <- as.integer(1 + with(dd, (time +1)/2))
tab(dd, ~ occ)
```

```
occ
    1     2 Total
10000 10000 20000
```

```
fit_gls <- gls(y ~ 1 + time, dd,
           weights = varIdent(form = ~ 1 | occ),
           correlation = corSymm(form = ~ occ | id))
summary(fit_gls)
```

```
Generalized least squares fit by REML
  Model: y ~ 1 + time
  Data: dd
       AIC      BIC    logLik
  76337.44 76376.96 -38163.72

Correlation Structure: General
 Formula: ~occ | id
 Parameter estimate(s):
 Correlation:
  1
2 0.353
Variance function:
 Structure: Different standard deviations per stratum
 Formula: ~1 | occ
 Parameter estimates:
       1        2
1.000000 1.405966
```

```
Coefficients:
                Value  Std.Error   t-value p-value
(Intercept) 1.005741 0.01416278  71.01298       0
time        1.993950 0.01001272 199.14170       0

 Correlation: 
     (Intr)
time 0.348 

Standardized residuals:
         Min           Q1          Med           Q3          Max 
-4.235190855 -0.685244335  0.002976327  0.675360721  3.740693626 

Residual standard error: 1.421711 
Degrees of freedom: 20000 total; 19998 residual
```

```
getV(fit_gls)
```

```
Marginal variance covariance matrix
       [,1]   [,2]
[1,] 2.0213 1.0033
[2,] 1.0033 3.9955
  Standard Deviations: 1.4217 1.9989 
```

```
getR(fit_gls)
```

```
Marginal variance covariance matrix
       [,1]   [,2]
[1,] 2.0213 1.0033
[2,] 1.0033 3.9955
  Standard Deviations: 1.4217 1.9989 
```

```
getG(fit_gls)
```

```
Marginal variance covariance matrix
       [,1]   [,2]
[1,] 2.0213 1.0033
[2,] 1.0033 3.9955
  Standard Deviations: 1.4217 1.9989 
```

```
#
```

Note: This approach allows you to fit an identifiably parametrized model on repeated measures data with a full multivariate variance matrix.