**Supplemental to**

**ESTABLISHING A TWIN REGISTERAn invaluable resource for (behavior) genetic,**

**epidemiological, biomarker and ‘omics’ studies by** Odintsova V, Willemsen G, Dolan CV,

Hottenga JJ, Martin NG, Slagboom PE, Ordoñana JR , Boomsma DI

Processing biological material in batches may give rise to batch effects, i.e., intra-batch correlation greater than zero. While there is an extensive literature on handling batch effects, we are not aware of any results addressing batch effects in twin data. A question that is specific to the twin design (or indeed any other design with naturally clustered observations) concerns the manner of allocation of twins to batches. One may allocate randomly by individual twin, or randomly by twin pair. The latter implies that the twin pairs share the batch, the former does not rule out batch sharing. The following are the results of simulation studies carried out to answer this question in three situations.

**How to allocate twins to batch in assay of metabolites in an extremely discordant and concordant (EDAC) twin design?**

Discordant and concordant twin pairs are selected on the basis of a phenotypic scores, for example aggression scores, for a biomarker study. Assays on the twins' urine samples are done to measure metabolites. The aim is to determine the association between metabolite levels and aggression. The metabolites are determined on plates, which means that the data are processed in batches. The present question concerns the allocation of twins to batch, given that plate is a source of systematic variation:

1) assign twin pairs randomly to batches

2) assign twin members (individuals) randomly to batches.

An additional question, specific to the EDAC design, is the choice of the independent variable. As the selection is on aggression scores it is statistically expedient to regress metabolite (predictor) on aggression (dependent). Selection on the predictor does not affect the regression, and if the selection is based on an EDAC scheme, the selection results in little loss of power. Alternatively one may choose to regress metabolite on the binary aggression scores (e.g., 0=low, 1=high). Regression on the continuous score is expected to confer greater power.

We make the following assumptions concerning the analysis. We assume that the twin data are to be analyzed in a single statistical model, which will include the discordant and concordant twins. With respect to this model, in testing the association of metabolite and aggression, we have to accommodate 1) the inherent two-level structure (family clustering of twins in twin pairs), and 2) the batch effects.

We consider two models:

1) Linear mixed model, in which effect of batch is accommodated by means of a random effect (variance component).

2) Fixed regression model with metabolite corrected for batch in one or two step procedure. Two step procedure: regress metabolite on plate first, use residuals in regression on predictor. One step procedure: regress metabolite on plate and on predictor at the same time.

The association between metabolite and aggression is accommodated by means of a fixed effect, i.e., regression of metabolite on binary (0/1) or continuous aggression score.

**Simulation 1. Random effects model.**

**Settings in the R code:**

NMZtot=600

r2=.05

h2m=.6

h2ag=.5

nplate=70

platevar=.25

selh=.0 # std units of aggression

sell=.0 # std units

nrep=50

The metabolite explains 5% (r2=.05) of the variance in aggression. The heritability of metabolite is .6 (h2m=.5), the heritability of the residual of aggression is .5 (h2ag=.5). The number of batches is 70, the number of twin pairs is 600 (NMZtot=600). The true phenotypic variances of metabolite and aggression are both set to equal 1 and the variance is .25 (platevar=.25). All variables have zero mean. So the metabolite variance is 1.25. The selection is based on a mean split (high aggr>0; low aggr<0). The number of replications is 50. All 600 pairs are in the analysis. Estimation method is by restricted ML (REML).

**Results:**

b1 is the parameter of interest (the regression coefficient in regression of metabolite on predictor). The t-statistic of b1 is the criterion to assess the effect of allocation on the power. The values in the tables 1-I to I-IV to are the means and standard deviations of the parameter b1, the standard error of b1 [se(b1)], the t-statistic (Wald test of b1), the conditional shared variance (s2A), the batch variance (s2plate) and the residual variance (s2E). Note that s2A and s2E are the variance components in a AE twin model. Table I-V contains the true parameter values in the continuous outcome phenotype case .

**Table 1-I: outcome binary (0/1), allocation by twin pairs**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | b1 | se(b1) | t-stat | s2A | s2plate | s2E |
| mean | 0.273 | 0.052 | **5.281** | 0.570 | 0.235 | 0.387 |
| stdv | 0.058 | 0.001 | 1.070 | 0.054 | 0.055 | 0.023 |

**Table 1-II: outcome binary (0/1), allocation by individual twins**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | b1 | se(b1) | t-stat | s2A | s2plate | s2E |
| mean | 0.279 | 0.052 | **5.321** | 0.571 | 0.258 | 0.386 |
| stdv | 0.058 | 0.001 | 1.044 | 0.048 | 0.052 | 0.023 |

**Table 1-III: outcome continuous, allocation by twin pairs**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | b1 | se(b1) | t-stat | s2A | s2plate | s2E |
| mean | 0.199 | 0.027 | **7.264** | 0.560 | 0.237 | 0.379 |
| stdv | 0.028 | 0.001 | 0.991 | 0.053 | 0.053 | 0.023 |

**Table 1-IV: outcome continuous, allocation by individual twins**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | b1 | se(b1) | t-stat | s2A | s2plate | s2E |
| mean | 0.200 | 0.028 | **7.265** | 0.562 | 0.258 | 0.378 |
| stdv | 0.027 | 0.001 | 0.960 | 0.048 | 0.052 | 0.022 |

**Table 1-V: expected (population) values (outcome continuous)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | variance components | | |
|  | b1 | se(b1) | t-stat | s2A | s2plate | s2E |
|  | .223 |  |  | .60 | .25 | .40 |

**Conclusion:**

Allocation regime (pairs vs. individuals) has no effect on the estimate of the parameter of interest (b1). We note that, as expected, regression on continuous predictors confers more power than regression on binary predictor (0/1). The difference in t-statistic is considerable (~7.26/5.3=~1.369). The variance components (A and E and Batch) appear to be slightly downwards biased in the based on allocation by pair, but accurate in the allocation by individual.

**Simulation 2: fixed batch effects in two steps or one step.**

It may be expedient to carry out analyses in two steps, i.e., first correct for batch effects, and second carry out the analysis of actual interest. We compare one and two step analyses in simulation 2. We used linear mixed modeling in simulation 1 (estimating the twin covariance conditional on predictor and batch). Here we use GEE, i.e., we correct the standard errors after the analyses using a sandwich correction. That is:

One step: using GEE regress metabolite on predictor and batch simultaneously.

Two step: first correct metabolite for batch effect and then use GEE to analyze the residuals

**Settings:**

Identical to simulation 1.

**Results:**

Again b1 is the parameter of interest (the regression coefficient in regression of metabolite on predictor). The t-statistic of b1 is the criterion to assess the effect of allocation on the power. Table 2 (I to IV) contain the results of the one-step procedure.

**Table 2: two step (first correct for plate, then regress on continuous or binary predictor)**

**Table 2-I: outcome binary (0/1), allocation by twin pairs**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.266 | 0.050 | **5.339** |
| stdv | 0.056 | 0.002 | 1.107 |

**Table 2-II: outcome binary (0/1), allocation by individuals**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.265 | 0.051 | **5.217** |
| stdv | 0.056 | 0.002 | 1.085 |

**Table 2-III: outcome continuous, allocation by twin pairs**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.190 | 0.026 | 7.277 |
| stdv | 0.027 | 0.002 | 1.113 |

**Table 2-IV: outcome continuous, allocation by individuals**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.189 | 0.027 | 7.103 |
| stdv | 0.027 | 0.002 | 1.078 |

Table 3 (I to IV) contain the results of the two-step procedure.

**Table 3: one step (regress on continuous or binary predictor and plate)**

**Table 3-I: outcome binary (0/1), allocation by twin pairs**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.273 | 0.051 | 5.335 |
| stdv | 0.059 | 0.002 | 1.128 |

**Table 3-II: outcome binary (0/1), allocation by individuals**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.269 | 0.051 | 5.269 |
| stdv | 0.058 | 0.002 | 1.136 |

**Table 3-III: outcome continuous, allocation by twin pairs**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.199 | 0.027 | 7.386 |
| stdv | 0.029 | 0.002 | 1.143 |

**Table 3-IV: outcome continuous, allocation by twin pairs**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.197 | 0.027 | 7.322 |
| stdv | 0.027 | 0.002 | 1.102 |

**Conclusions:**

Conclusions are the same as those based on Simulation 1. Allocation regime (pairs vs. individuals) has little effect on the test of the parameter of interest. Again, as expected, regression of continuous predictors confers more power that regression on binary predictor (0/1). The difference in z-statistic is considerable. Finally we see little differences between one and two step procedure.

**Simulation 3: More extreme selection and fixed plate effects in two step or one step.**

This simulation is the same as simulation 2. However here we employ a more extreme selection criterion. Rather than a mean split (Simulations 1 and 2), the selection of high and low scoring twins is on the basis of the criteria > .5 std unit or < -.5 std units. As in simulation 2, we carry out one and two step analyses using GEE. Given the selection, we set the total sample size to 5000 (random sample) and select from this sample based on the criteria mentioned.

**Settings:**

NMZtot=5000

r2=.05

h2m=.6

h2ag=.5

nplate=70

platevar=.25

selh=.5 # std units of aggression (var=1)

sell=-.5 # std units

nrep=50

**Results:**

The parameter b1 is the parameter of interest (the regression coefficient in regression of metabolite on predictor). The t-statistic of b1 is the criterion to assess the effect of allocation on the power. Tables 4 I to IV contains the results of the two step procedure, and table 5 I to IV contain the results of the one-step procedure.

**Table 4: two step (first correct for plate, then regress on continuous or binary predictor)**

**Table 4-I: outcome binary (0/1), allocation by twin pairs**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.465 | 0.032 | 14.681 |
| stdv | 0.026 | 0.001 | 0.854 |

**Table 4-II: outcome binary (0/1), allocation by individuals**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.468 | 0.032 | 14.713 |
| stdv | 0.025 | 0.001 | 0.829 |

**Table 4-III: outcome continuous, allocation by twin pairs**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.208 | 0.012 | 17.055 |
| stdv | 0.011 | 0.000 | 0.954 |

**Table 4-IV: outcome continuous, allocation by individuals**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.209 | 0.012 | 17.061 |
| stdv | 0.011 | 0.000 | 0.917 |

**Table 5: one step (regress on continuous or binary predictor and plate)**

**Table 5-I: outcome binary (0/1), allocation by twin pairs**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.472 | 0.032 | 14.762 |
| stdv | 0.026 | 0.001 | 0.864 |

**Table 5-II: outcome binary (0/1), allocation by individuals**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.473 | 0.032 | 14.805 |
| stdv | 0.026 | 0.001 | 0.856 |

**Table 5-III: outcome continuous, allocation by twin pairs**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.212 | 0.012 | 17.212 |
| stdv | 0.011 | 0.000 | 0.947 |

**Table 5-IV: outcome continuous, allocation by twin pairs**

|  |  |  |  |
| --- | --- | --- | --- |
|  | b1 | se(b1) | z-stat (robust) |
| mean | 0.212 | 0.012 | 17.240 |
| stdv | 0.011 | 0.000 | 0.929 |

**Conclusions:**

The conclusions are consistent with those of simulations 1 and 2. The allocation regime, i.e., pairs vs. individuals, has little effect on the test of the parameter of interest. The regression of continuous predictors confers more power that regression on binary predictor (0/1), as expected. The difference in z-statistic is considerable. There is little difference between the results of the one and two step procedure.

**How to allocate twins to batches in assay of metabolites in the classical twin design (ACE model)**

**Simulation 4:** estimating ACE variance components in the classical twin design.

In simulation 1, we noted that batch allocation by twin pair appeared to result in a slight bias in the estimates of the variance components. In simulation 4, we examine the effect on variance components by fitting an ACE model to twin data. Here we treat batch as a random effect and as a fixed effects, and we carry out both one step and two step analyses. We consider relatively small sample sizes. We use linear mixed modeling with REML estimation.

The settings and results are given in the following table 6.

**Table 6: variance components estimated in the classical twin design**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 step batch random | | | | 1 step batch fixed | | | 2 step (batch random) | | | 2 step (batch fixed) | | |
|  | rvB | rvA | rvC | rvE | fvA | fvC | fvE | crvA | crvC | crvE | cfvA | cfvC | cfvE |
| A | 1 | 4 | 2 | 4 | 4 | 2 | 4 | 4 | 2 | 4 | 4 | 2 | 4 |
| B | 0 | *3.957* | *2.068* | *3.991* | *3.957* | *2.068* | *3.991* | *3.957* | *2.068* | *3.991* | *3.957* | *2.068* | *3.991* |
| C | 0 | *3.881* | *2.093* | *3.964* | *3.881* | *2.093* | *3.964* | *3.881* | *2.093* | *3.964* | *3.881* | *2.093* | *3.964* |
| **1p** | **1.014** | **3.719** | **2.264** | **4.011** | **3.720** | **2.261** | **4.011** | **3.684** | **2.014** | **4.015** | **3.666** | **1.872** | **4.017** |
| **2i** | **0.997** | **4.052** | **2.044** | **3.991** | **4.039** | **2.051** | **3.996** | **4.046** | **2.047** | **3.825** | **4.029** | **2.056** | **3.788** |
| **3p** | **0.997** | **3.851** | **2.132** | **4.017** | **3.846** | **2.136** | **4.017** | **3.821** | **1.867** | **4.018** | **3.811** | **1.718** | **4.021** |
| **4i** | **1.00** | **3.910** | **2.034** | **4.035** | **3.910** | **2.041** | **4.036** | **3.91** | **2.034** | **3.861** | **3.90** | **2.041** | **3.821** |
| **5p** | **1.001** | **3.943** | **2.034** | **4.014** | **3.953** | **2.026** | **4.013** | **3.906** | **1.673** | **4.018** | **3.864** | **1.374** | **4.023** |
| **6i** | **0.996** | **3.834** | **2.132** | **4.023** | **3.879** | **2.095** | **4.015** | **3.823** | **2.135** | **3.777** | **3.864** | **2.098** | **3.670** |

All cases: var(A) = 4 (\_vA); var(C) = 2 (\_vC); var(E) = 4 (\_vE); var(Batch)=1 (\_vB); Number of replications = 250

1 step analyses:

rvB, rvA, rvC, rvE: variance components all random effects

fVA, fvC, fvE: batch fixed effects, A, V and E variance components random effects

2 step analyses:

crvA, crvC, crvE; variance components after correction for random batch effects

cfvA, cfvC, cfvE; variance components after correction for fixed batch effects

A. true values

B. Nmz=Ndz=200; no batch effects and no batch effect modeled, i.e., standard ACE variance component model

C. Nmz=Ndz=120; no batch effects and no batch effect modeled, i.e., standard ACE variance component model

1p. Nmz=Ndz=120;Nbatch=15; random assignment by pair (twin share batch)

2i. Nmz=Ndz=120;Nbatch=15; random assignment by individual

3p. Nmz=Ndz=200;Nbatch=25; random assignment by pair (twin share batch)

4i. Nmz=Ndz=200;Nbatch=25; random assignment by individual

5p. Nmz=Ndz=200;Nbatch=40; random assignment by pair (twin share batch)

6i. Nmz=Ndz=200;Nbatch=40; random assignment by individual

**Conclusion:**

The results demonstrate that allocation regime has little effect in the one step analyses, regardless of whether this is based on random effects or fixed effects modeling of batch. In the two step procedures, we note a downward bias in the estimates of the A and C variance components. This bias is greater in given the allocation by twin pairs, and greater as the number of batches increases (compare 15, 25, and 40 batch conditions).

**Appendices: R code.**

**Appendix 1. Code: fixed effect, one and two step procedure**

rm(list=ls(all=TRUE))

seed=201

set.seed(seed)

#

library(MASS)

library(gee)

#library(nlme)

# fiona

exact=FALSE

NMZtot=5000

#

#

MLmethod='REML'

r2=.05

h2m=.6

h2ag=.5

nplate=70

platevar=.25

selh=.5 # std units of agressie (var=1)

sell=-.5 # std units

#

nrep=50

#

#

#

Slat=matrix(c(

1,0,0,0,1,0,0,0, # AM1

0,1,0,0,0,0,0,0, # EM1

0,0,1,0,0,0,1,0, # AAg1

0,0,0,1,0,0,0,0, # EAg1

1,0,0,0,1,0,0,0, # AM2

0,0,0,0,0,1,0,0, # EM2

0,0,1,0,0,0,1,0, # AAg2

0,0,0,0,0,0,0,1 # EAg2

),8,8,byrow=T)

mlat=rep(0,8)

#

platesd=sqrt(platevar)

b=sqrt(r2)

resv=1-r2

sdv=sqrt(resv)

#

#

results3a=matrix(0,nrep,5)

results4a=matrix(0,nrep,5)

results5a=matrix(0,nrep,5)

results4b=matrix(0,nrep,5)

results5b=matrix(0,nrep,5)

results4a2=matrix(0,nrep,5)

results5a2=matrix(0,nrep,5)

results4b2=matrix(0,nrep,5)

results5b2=matrix(0,nrep,5)

#

for (irep in 1:nrep) {

#

d1=mvrnorm(NMZtot,mu=mlat,Sigma=Slat,emp=exact)

d2=matrix(0,NMZtot,7)

d2[,1]=d1[,1]\*sqrt(h2m) + d1[,2]\*sqrt(1-h2m)

d2[,3]=d1[,5]\*sqrt(h2m) + d1[,6]\*sqrt(1-h2m)

res1=(d1[,3]\*sqrt(h2ag) + d1[,4]\*sqrt(1-h2ag))\*sdv

res2=(d1[,7]\*sqrt(h2ag) + d1[,8]\*sqrt(1-h2ag))\*sdv

d2[,2]=d2[,1]\*b+res1

d2[,4]=d2[,3]\*b+res2

d2=as.data.frame(d2)

# 1 2 3 4 5 6 7

colnames(d2)=c('mbol1','ag1','mbol2','ag2','sel','eff1','eff2')

#

summary(lm(ag1~mbol1,data=d2))

summary(lm(ag2~mbol2,data=d2))

#

sel=(d2$ag1 < d2$ag2)

d2[sel,1:5]=d2[sel,c(3,4,1,2,5)]

#

#

#

filt=matrix(0,NMZtot,3)

filt[,1]=cch=(d2$ag1>selh& d2$ag2>selh)

filt[,2]=ccl=(d2$ag1<sell & d2$ag2<sell)

filt[,3]=disc=((d2$ag1<sell & d2$ag2>selh) | (d2$ag2<sell & d2$ag1>selh))

d2[,5]='no'

d2[cch,5]='cchi'

d2[cch,6:7]=1

d2[ccl,5]='cclo'

d2[disc,5]='disc'

d2[disc,6]=1

#

d4=d5=d3=d2[d2[,5]!="no",]

#

N=dim(d3)[1]

#

# assign plates

#

nplc=sample(c(1:nplate),N,replace=T)

npl1=sample(c(1:nplate),N,replace=T)

npl2=sample(c(1:nplate),N,replace=T)

platec=rnorm(nplate,0,(platesd))

plates=rnorm(nplate,0,(platesd))

if (exact) {

platec=(platec-mean(platec))/sd(platec)

plates=(plates-mean(plates))/sd(plates)

}

#

# shared plate

d4[,1]=d4[,1]+platec[nplc]

d4[,3]=d4[,3]+platec[nplc]

# random assignment

d5[,1]=d5[,1]+plates[npl1]

d5[,3]=d5[,3]+plates[npl2]

#

#colnames(d2)=c('mbol1','ag1','mbol2','ag2','sel')

#

pair=rep(1:N,each=2)

twin=rep(c(1:2),N)

d5L=as.data.frame(matrix(0,N\*2,7+2))

d4L=as.data.frame(matrix(0,N\*2,7+2))

d3L=as.data.frame(matrix(0,N\*2,7))

#

d3L[,1]=d4L[,1]=d5L[,1]=pair # pair

d3L[,2]=d4L[,2]=d5L[,2]=twin # member

d3L[,7]=d4L[,7]=d5L[,7]=1 # unit

d3L[,3]=d3[pair,5]

d4L[,3]=d3[pair,5]

d5L[,3]=d3[pair,5]

#

indi=matrix(c(1,2,3,4),2,2,byrow=T)

#

print(' start H->V ')

ii=0

for (i in 1:N) {

for (j in 1:2) {

ii=ii+1

# ---------------------------------------------------3

d3L[ii,6]=d3[i,5+j]

d3L[ii,4]=d3[i,indi[j,1]] # mbol1

d3L[ii,5]=d3[i,indi[j,2]] # agr1

# ------------------------------------------------- 4

d4L[ii,6]=d3[i,5+j]

d4L[ii,4]=d4[i,indi[j,1]] # mbol1

d4L[ii,5]=d4[i,indi[j,2]] # agr1

if (j==1) {

d4L[ii,8]=nplc[i]

if (d4[i,5]=='disc') {d4L[ii,6]=1}

}

if (j==2) {

d4L[ii,8]=nplc[i]

}

# --------------------------------------------------- 5

d5L[ii,6]=d3[i,5+j]

d5L[ii,4]=d5[i,indi[j,1]] # mbol1

d5L[ii,5]=d5[i,indi[j,2]] # agr1

if (j==1) {

d5L[ii,8]=npl1[i]

if (d5[i,5]=='disc') {d5L[ii,6]=1}

}

if (j==2) {

d5L[ii,8]=npl2[i]

}

#

}} # twin pair

print(' end H->V ')

#

tagpl=c('plate','mbolc')

vnames=c('pair','mem','type','mbol','ag','eff','unit')

colnames(d3L)=vnames

colnames(d4L)=c(vnames,tagpl)

colnames(d5L)=c(vnames,tagpl)

#

#

correct4=lm(mbol~as.factor(plate),data=d4L)

d4L[,9]=correct4$residuals

correct5=lm(mbol~as.factor(plate),data=d5L)

d5L[,9]=correct5$residuals

r3a=gee(mbol~eff,id=pair,corstr='exchangeable',data=d3L)

r4a=gee(mbolc~eff,id=pair,corstr='exchangeable',data=d4L)

r5a=gee(mbolc~eff,id=pair,corstr='exchangeable',data=d5L)

r4a2=gee(mbol~eff+as.factor(plate),id=pair,corstr='exchangeable',data=d4L)

r5a2=gee(mbol~eff+as.factor(plate),id=pair,corstr='exchangeable',data=d5L)

r3b=gee(mbol~ag,id=pair,corstr='exchangeable',data=d3L)

r4b=gee(mbolc~ag,id=pair,corstr='exchangeable',data=d4L)

r5b=gee(mbolc~ag,id=pair,corstr='exchangeable',data=d5L)

r4b2=gee(mbol~ag+as.factor(plate),id=pair,corstr='exchangeable',data=d4L)

r5b2=gee(mbol~ag+as.factor(plate),id=pair,corstr='exchangeable',data=d5L)

results3a[irep,1:5]=as.numeric(summary(r3a)$coefficients[2,])

results4a[irep,1:5]=as.numeric(summary(r4a)$coefficients[2,])

results5a[irep,1:5]=as.numeric(summary(r5a)$coefficients[2,])

results4a2[irep,1:5]=as.numeric(summary(r4a2)$coefficients[2,])

results5a2[irep,1:5]=as.numeric(summary(r5a2)$coefficients[2,])

results4b[irep,1:5]=as.numeric(summary(r4b)$coefficients[2,])

results5b[irep,1:5]=as.numeric(summary(r5b)$coefficients[2,])

results4b2[irep,1:5]=as.numeric(summary(r4b2)$coefficients[2,])

results5b2[irep,1:5]=as.numeric(summary(r5b2)$coefficients[2,])

}

mrep=nrep

round(apply(results3a[1:mrep,c(1,4,5)],2,mean),3) # eff=0/1 random allo of pairs

round(apply(results4a[1:mrep,c(1,4,5)],2,mean),3) # eff=0/1 random allo of pairs

round(apply(results4a[1:mrep,c(1,4,5)],2,sd),3) # eff=0/1 random allo of pairs

round(apply(results5a[1:mrep,c(1,4,5)],2,mean),3) # eff=0/1 random allo of individuals

round(apply(results5a[1:mrep,c(1,4,5)],2,sd),3) # eff=0/1 random allo of individuals

round(apply(results4b[1:mrep,c(1,4,5)],2,mean),3) # eff=ag

round(apply(results4b[1:mrep,c(1,4,5)],2,sd),3) # eff=ag

round(apply(results5b[1:mrep,c(1,4,5)],2,mean),3) # eff=ag

round(apply(results5b[1:mrep,c(1,4,5)],2,sd),3) # eff=ag

round(apply(results4a2[1:mrep,c(1,4,5)],2,mean),3) # eff=0/1 random allo of pairs

round(apply(results4a2[1:mrep,c(1,4,5)],2,sd),3) # eff=0/1 random allo of pairs

round(apply(results5a2[1:mrep,c(1,4,5)],2,mean),3) # eff=0/1 random allo of individuals

round(apply(results5a2[1:mrep,c(1,4,5)],2,sd),3) # eff=0/1 random allo of individuals

round(apply(results4b2[1:mrep,c(1,4,5)],2,mean),3) # eff=ag

round(apply(results4b2[1:mrep,c(1,4,5)],2,sd),3) # eff=ag

round(apply(results5b2[1:mrep,c(1,4,5)],2,mean),3) # eff=ag

round(apply(results5b2[1:mrep,c(1,4,5)],2,sd),3) # eff=ag

**Appendix 2. Code: random effects model**

rm(list=ls(all=TRUE))

seed=301

set.seed(seed)

#

library(MASS)

#library(gee)

library(nlme)

# fiona

exact=FALSE

NMZtot=1000

#

#

MLmethod='REML'

r2=.05

h2m=.6

h2ag=.5

nplate=70

platevar=.25

selh=.25 # std units of agressie (var=1)

sell=-.25 # std units

#

nrep=100

#

#

#

Slat=matrix(c(

1,0,0,0,1,0,0,0, # AM1

0,1,0,0,0,0,0,0, # EM1

0,0,1,0,0,0,1,0, # AAg1

0,0,0,1,0,0,0,0, # EAg1

1,0,0,0,1,0,0,0, # AM2

0,0,0,0,0,1,0,0, # EM2

0,0,1,0,0,0,1,0, # AAg2

0,0,0,0,0,0,0,1 # EAg2

),8,8,byrow=T)

mlat=rep(0,8)

#

platesd=sqrt(platevar)

b=sqrt(r2)

resv=1-r2

sdv=sqrt(resv)

#

#

results4a=matrix(0,nrep,8)

results5a=matrix(0,nrep,8)

results4b=matrix(0,nrep,8)

results5b=matrix(0,nrep,8)

#

for (irep in 1:nrep) {

#

d1=mvrnorm(NMZtot,mu=mlat,Sigma=Slat,emp=exact)

d2=matrix(0,NMZtot,7)

d2[,1]=d1[,1]\*sqrt(h2m) + d1[,2]\*sqrt(1-h2m)

d2[,3]=d1[,5]\*sqrt(h2m) + d1[,6]\*sqrt(1-h2m)

res1=(d1[,3]\*sqrt(h2ag) + d1[,4]\*sqrt(1-h2ag))\*sdv

res2=(d1[,7]\*sqrt(h2ag) + d1[,8]\*sqrt(1-h2ag))\*sdv

d2[,2]=d2[,1]\*b+res1

d2[,4]=d2[,3]\*b+res2

d2=as.data.frame(d2)

# 1 2 3 4 5 6 7

colnames(d2)=c('mbol1','ag1','mbol2','ag2','sel','eff1','eff2')

#

summary(lm(ag1~mbol1,data=d2))

summary(lm(ag2~mbol2,data=d2))

#

sel=(d2$ag1 < d2$ag2)

d2[sel,1:5]=d2[sel,c(3,4,1,2,5)]

#

#

#

filt=matrix(0,NMZtot,3)

filt[,1]=cch=(d2$ag1>selh& d2$ag2>selh)

filt[,2]=ccl=(d2$ag1<sell & d2$ag2<sell)

filt[,3]=disc=((d2$ag1<sell & d2$ag2>selh) | (d2$ag2<sell & d2$ag1>selh))

d2[,5]='no'

d2[cch,5]='cchi'

d2[cch,6:7]=1

d2[ccl,5]='cclo'

d2[disc,5]='disc'

d2[disc,6]=1

#

d4=d5=d3=d2[d2[,5]!="no",]

#

N=dim(d3)[1]

#

# assign plates

#

nplc=sample(c(1:nplate),N,replace=T)

npl1=sample(c(1:nplate),N,replace=T)

npl2=sample(c(1:nplate),N,replace=T)

platec=rnorm(nplate,0,(platesd))

plates=rnorm(nplate,0,(platesd))

if (exact) {

platec=(platec-mean(platec))/sd(platec)

plates=(plates-mean(plates))/sd(plates)

}

#

# shared plate

d4[,1]=d4[,1]+platec[nplc]

d4[,3]=d4[,3]+platec[nplc]

# random assignment

d5[,1]=d5[,1]+plates[npl1]

d5[,3]=d5[,3]+plates[npl2]

#

#colnames(d2)=c('mbol1','ag1','mbol2','ag2','sel')

#

pair=rep(1:N,each=2)

twin=rep(c(1:2),N)

d5L=as.data.frame(matrix(0,N\*2,7+nplate))

d4L=as.data.frame(matrix(0,N\*2,7+nplate))

d3L=as.data.frame(matrix(0,N\*2,7))

d3L[,1]=d4L[,1]=d5L[,1]=pair # pair

d3L[,2]=d4L[,2]=d5L[,2]=twin # member

d3L[,7]=d4L[,7]=d5L[,7]=1 # unit

d3L[,3]=d3[pair,5]

d4L[,3]=d3[pair,5]

d5L[,3]=d3[pair,5]

#

indi=matrix(c(1,2,3,4),2,2,byrow=T)

#

print(' start H->V ')

ii=0

for (i in 1:N) {

for (j in 1:2) {

ii=ii+1

# ---------------------------------------------------3

d3L[ii,6]=d3[i,5+j]

d3L[ii,4]=d3[i,indi[j,1]] # mbol1

d3L[ii,5]=d3[i,indi[j,2]] # agr1

# ------------------------------------------------- 4

d4L[ii,6]=d3[i,5+j]

d4L[ii,4]=d4[i,indi[j,1]] # mbol1

d4L[ii,5]=d4[i,indi[j,2]] # agr1

if (j==1) {

plate=rep(0,nplate)

plate[nplc[i]]=1

d4L[ii,8:(7+nplate)]=plate

if (d4[i,5]=='disc') {d4L[ii,6]=1}

}

if (j==2) {

plate=rep(0,nplate)

plate[nplc[i]]=1

d4L[ii,8:(7+nplate)]=plate

}

# --------------------------------------------------- 5

d5L[ii,6]=d3[i,5+j]

d5L[ii,4]=d5[i,indi[j,1]] # mbol1

d5L[ii,5]=d5[i,indi[j,2]] # agr1

if (j==1) {

plate=rep(0,nplate)

plate[npl1[i]]=1

d5L[ii,8:(7+nplate)]=plate

if (d5[i,5]=='disc') {d5L[ii,6]=1}

}

if (j==2) {

plate=rep(0,nplate)

plate[npl2[i]]=1

d5L[ii,8:(7+nplate)]=plate

}

#

}} # twin pair

print(' end H->V ')

tagpl=paste('pl',1:nplate,sep="")

vnames=c('pair','mem','type','mbol','ag','eff','unit')

colnames(d3L)=vnames

colnames(d4L)=c(vnames,tagpl)

colnames(d5L)=c(vnames,tagpl)

#

#

control=lmeControl(maxIter = 5000, msMaxIter = 5000, tolerance = 1e-5, niterEM = 2500,

msMaxEval = 5000,

msTol = 1e-6)

# r3=lme(mbol~1+ag,random=~1|pair,control=control,data=d3L)

#

r4a=lme(mbol~1+eff,

random=list(

unit=(pdIdent(~pl1+pl2+pl3+pl4+pl5+pl6+pl7+pl8+pl9+pl10+

pl11+pl12+pl13+pl14+pl15+pl16+pl17+pl18+pl19+pl20+

pl21+pl22+pl23+pl24+pl25+pl26+pl27+pl28+pl29+pl30+

pl31+pl32+pl33+pl34+pl35+pl36+pl37+pl38+pl39+pl40+

pl41+pl42+pl43+pl44+pl45+pl46+pl47+pl48+pl49+pl50+

pl51+pl52+pl53+pl54+pl55+pl56+pl57+pl58+pl59+pl60+

pl61+pl62+pl63+pl64+pl65+pl66+pl67+pl68+pl69+pl70-1)),

pair=(pdDiag(~unit-1))),

control=control,data=d4L,method=MLmethod)

r5a=lme(mbol~1+eff,

random=list(

unit=(pdIdent(~pl1+pl2+pl3+pl4+pl5+pl6+pl7+pl8+pl9+pl10+

pl11+pl12+pl13+pl14+pl15+pl16+pl17+pl18+pl19+pl20+

pl21+pl22+pl23+pl24+pl25+pl26+pl27+pl28+pl29+pl30+

pl31+pl32+pl33+pl34+pl35+pl36+pl37+pl38+pl39+pl40+

pl41+pl42+pl43+pl44+pl45+pl46+pl47+pl48+pl49+pl50+

pl51+pl52+pl53+pl54+pl55+pl56+pl57+pl58+pl59+pl60+

pl61+pl62+pl63+pl64+pl65+pl66+pl67+pl68+pl69+pl70-1)),

pair=(pdDiag(~unit-1))),

control=control,data=d5L,method=MLmethod)

#

r4b=lme(mbol~1+ag,

random=list(

unit=(pdIdent(~pl1+pl2+pl3+pl4+pl5+pl6+pl7+pl8+pl9+pl10+

pl11+pl12+pl13+pl14+pl15+pl16+pl17+pl18+pl19+pl20+

pl21+pl22+pl23+pl24+pl25+pl26+pl27+pl28+pl29+pl30+

pl31+pl32+pl33+pl34+pl35+pl36+pl37+pl38+pl39+pl40+

pl41+pl42+pl43+pl44+pl45+pl46+pl47+pl48+pl49+pl50+

pl51+pl52+pl53+pl54+pl55+pl56+pl57+pl58+pl59+pl60+

pl61+pl62+pl63+pl64+pl65+pl66+pl67+pl68+pl69+pl70-1)),

pair=(pdDiag(~unit-1))),

control=control,data=d4L,method=MLmethod)

#

r5b=lme(mbol~1+ag,

random=list(

unit=(pdIdent(~pl1+pl2+pl3+pl4+pl5+pl6+pl7+pl8+pl9+pl10+

pl11+pl12+pl13+pl14+pl15+pl16+pl17+pl18+pl19+pl20+

pl21+pl22+pl23+pl24+pl25+pl26+pl27+pl28+pl29+pl30+

pl31+pl32+pl33+pl34+pl35+pl36+pl37+pl38+pl39+pl40+

pl41+pl42+pl43+pl44+pl45+pl46+pl47+pl48+pl49+pl50+

pl51+pl52+pl53+pl54+pl55+pl56+pl57+pl58+pl59+pl60+

pl61+pl62+pl63+pl64+pl65+pl66+pl67+pl68+pl69+pl70-1)),

pair=(pdDiag(~unit-1))),

control=control,data=d5L,method=MLmethod)

#

par\_fixed5a=summary(r5a)$tTable

par\_sd5a=exp(as.numeric(attr(r5a$apVar,"Pars")))

par\_var5a=par\_sd5a^2

#

par\_fixed5b=summary(r5b)$tTable

par\_sd5b=exp(as.numeric(attr(r5b$apVar,"Pars")))

par\_var5b=par\_sd5b^2

#

par\_fixed4a=summary(r4a)$tTable

par\_sd4a=exp(as.numeric(attr(r4a$apVar,"Pars")))

par\_var4a=par\_sd4a^2

#

par\_fixed4b=summary(r4b)$tTable

par\_sd4b=exp(as.numeric(attr(r4b$apVar,"Pars")))

par\_var4b=par\_sd4b^2

#

results4a[irep,1:5]=par\_fixed4a[2,]

results4a[irep,6:8]=par\_var4a

results4b[irep,1:5]=par\_fixed4b[2,]

results4b[irep,6:8]=par\_var4b

#

results5a[irep,1:5]=par\_fixed5a[2,]

results5a[irep,6:8]=par\_var5a

results5b[irep,1:5]=par\_fixed5b[2,]

results5b[irep,6:8]=par\_var5b

print(c(irep,nrep))

print(par\_fixed4a)

print(par\_var4a)

}

mrep=50

round(apply(results4a[1:mrep,c(1,2,4,6,7,8)],2,mean),3) # eff=0/1 random allo of pairs

round(apply(results4a[1:mrep,c(1,2,4,6,7,8)],2,sd),3) # eff=0/1 random allo of pairs

round(apply(results5a[1:mrep,c(1,2,4,6,7,8)],2,mean),3) # eff=0/1 random allo of individuals

round(apply(results5a[1:mrep,c(1,2,4,6,7,8)],2,sd),3) # eff=0/1 random allo of individuals

round(apply(results4b[1:mrep,c(1,2,4,6,7,8)],2,mean),3) # eff=ag

round(apply(results4b[1:mrep,c(1,2,4,6,7,8)],2,sd),3) # eff=ag

round(apply(results5b[1:mrep,c(1,2,4,6,7,8)],2,mean),3) # eff=ag

round(apply(results5b[1:mrep,c(1,2,4,6,7,8)],2,sd),3) # eff=ag

**Appendix 3. Code: ACE model no batch effect and no batch effects modeled.**

rm(list=ls(all=TRUE))

#

# random batch

#

library(MASS)

library(nlme)

#

control=lmeControl(maxIter = 5000, msMaxIter = 5000, tolerance = 1e-5, niterEM = 2500,

msMaxEval = 5000,

msTol = 1e-6)

#

MLmethod='REML'

exact=F

sharedB=F

#

Nmz=Ndz=120

Smz=matrix(c(

10,6,6,10),2,2)

Sdz=matrix(c(

10,4,4,10),2,2)

nbatch=40

vbatch=0

nrep=250

#

results=matrix(NA,nrep,13)

colnames(results)=c(

'rvb','rva','rvc','rve','fva','fvc','fve',

'crva','crvc','crve','cfva','cfvc','cfve')

batch=c(1:nbatch)

wr=F

#

for (irep in 1:nrep) {

print(irep)

#

b\_eff=scale(rnorm(nbatch,0,1))\*sqrt(vbatch)

#

bmz1=sample(batch,Nmz,replace=T)

bdz1=sample(batch,Ndz,replace=T)

bmz2=sample(batch,Nmz,replace=T)

bdz2=sample(batch,Ndz,replace=T)

if (sharedB) {

bmz2=bmz1

bdz2=bdz1

}

#

dmz=matrix(0,Nmz,7)

dmz[,1]=1

dmz[,2:3]=mvrnorm(Nmz,rep(0,2),Sigma=Smz,emp=exact)

dmz[,4]=bmz1

dmz[,5]=bmz2

dmz[,2]=dmz[,2]+b\_eff[bmz1]

dmz[,3]=dmz[,3]+b\_eff[bmz2]

#

ddz=matrix(0,Ndz,7)

ddz[,1]=2

ddz[,2:3]=mvrnorm(Ndz,rep(0,2),Sigma=Sdz,emp=exact)

ddz[,4]=bdz1

ddz[,5]=bdz2

ddz[,2]=ddz[,2]+b\_eff[bdz1]

ddz[,3]=ddz[,3]+b\_eff[bdz2]

#

keepdmz=dmz[,2:3]

keepddz=ddz[,2:3]

#

ddz1=matrix(0,Nmz\*2,11)

dmz1=matrix(0,Nmz\*2,11)

#

ii=0

for (i in 1:Nmz) {

for (j in 1:2) {

ii=ii+1

dmz1[ii,1]=i

dmz1[ii,2]=j

dmz1[ii,3]=1

dmz1[ii,4]=dmz[i,(1+j)]

dmz1[ii,5]=dmz[i,(3+j)]

dmz1[ii,6]=sqrt(.5)

dmz1[ii,7]=sqrt(.5)

dmz1[ii,9]=sqrt(.25)

dmz1[ii,10]=sqrt(.75)

}}

#

ii=0

for (i in 1:Ndz) {

for (j in 1:2) {

ii=ii+1

ddz1[ii,1]=i+Nmz

ddz1[ii,2]=j

ddz1[ii,3]=2

ddz1[ii,4]=ddz[i,(1+j)]

ddz1[ii,5]=ddz[i,(3+j)]

ddz1[ii,6]=sqrt(.5)

if (j==1) ddz1[ii,7]=sqrt(.5)

if (j==2) ddz1[ii,8]=sqrt(.5)

ddz1[ii,9]=sqrt(.25)

if (j==1) ddz1[ii,10]=sqrt(.75)

if (j==2) ddz1[ii,11]=sqrt(.75)

}}

#

twdat=rbind(dmz1,ddz1)

twdat=cbind(twdat,rep(1,(Nmz+Ndz)\*2))

#

colnames(twdat)=c('famnr','twnr','zyg','ph','batch','Ac','A1','A2','Dc','D1','D2','unit')

twdat=as.data.frame(twdat)

twdat$batch=as.factor(twdat$batch)

#

if (wr) write.table(twdat,file='twdat',row.names=F)

#

testf=tryCatch(lme(ph~1,

random=list(

famnr=pdIdent(~Ac+A1+A2-1),

famnr=pdIdent(~unit-1)

),control=control,data=twdat,method=MLmethod), error = function(e) {-999} )

#

contf=!is.numeric(testf)

if (contf) {

nrf=dim(VarCorr(testf))[1]

vcompf=na.omit(matrix(as.numeric((VarCorr(testf))),nrf,2,byrow=F))[c(1,4,5),1]

results[irep,1:3]=vcompf

}#

#

} # irep

**Appendix 4. Code: ACE model with batch effects: random effects model, fixed effects model, one and two step procedures.**

rm(list=ls(all=TRUE))

#

# random batch

#

library(MASS)

library(nlme)

#

control=lmeControl(maxIter = 5000, msMaxIter = 5000, tolerance = 1e-5, niterEM = 2500,

msMaxEval = 5000,

msTol = 1e-6)

#

MLmethod='REML'

exact=F

sharedB=T

#

Nmz=Ndz=200

Smz=matrix(c(

10,6,6,10),2,2)

Sdz=matrix(c(

10,4,4,10),2,2)

nbatch=25

vbatch=1

nrep=250

#

results=matrix(NA,nrep,13)

colnames(results)=c(

'rvb','rva','rvc','rve','fva','fvc','fve',

'crva','crvc','crve','cfva','cfvc','cfve')

batch=c(1:nbatch)

wr=F

#

for (irep in 1:nrep) {

print(irep)

#

b\_eff=scale(rnorm(nbatch,0,1))\*sqrt(vbatch)

#

bmz1=sample(batch,Nmz,replace=T)

bdz1=sample(batch,Ndz,replace=T)

bmz2=sample(batch,Nmz,replace=T)

bdz2=sample(batch,Ndz,replace=T)

if (sharedB) {

bmz2=bmz1

bdz2=bdz1

}

#

dmz=matrix(0,Nmz,7)

dmz[,1]=1

dmz[,2:3]=mvrnorm(Nmz,rep(0,2),Sigma=Smz,emp=exact)

dmz[,4]=bmz1

dmz[,5]=bmz2

dmz[,2]=dmz[,2]+b\_eff[bmz1]

dmz[,3]=dmz[,3]+b\_eff[bmz2]

#

ddz=matrix(0,Ndz,7)

ddz[,1]=2

ddz[,2:3]=mvrnorm(Ndz,rep(0,2),Sigma=Sdz,emp=exact)

ddz[,4]=bdz1

ddz[,5]=bdz2

ddz[,2]=ddz[,2]+b\_eff[bdz1]

ddz[,3]=ddz[,3]+b\_eff[bdz2]

#

keepdmz=dmz[,2:3]

keepddz=ddz[,2:3]

#

ddz1=matrix(0,Nmz\*2,11)

dmz1=matrix(0,Nmz\*2,11)

#

ii=0

for (i in 1:Nmz) {

for (j in 1:2) {

ii=ii+1

dmz1[ii,1]=i

dmz1[ii,2]=j

dmz1[ii,3]=1

dmz1[ii,4]=dmz[i,(1+j)]

dmz1[ii,5]=dmz[i,(3+j)]

dmz1[ii,6]=sqrt(.5)

dmz1[ii,7]=sqrt(.5)

dmz1[ii,9]=sqrt(.25)

dmz1[ii,10]=sqrt(.75)

}}

#

ii=0

for (i in 1:Ndz) {

for (j in 1:2) {

ii=ii+1

ddz1[ii,1]=i+Nmz

ddz1[ii,2]=j

ddz1[ii,3]=2

ddz1[ii,4]=ddz[i,(1+j)]

ddz1[ii,5]=ddz[i,(3+j)]

ddz1[ii,6]=sqrt(.5)

if (j==1) ddz1[ii,7]=sqrt(.5)

if (j==2) ddz1[ii,8]=sqrt(.5)

ddz1[ii,9]=sqrt(.25)

if (j==1) ddz1[ii,10]=sqrt(.75)

if (j==2) ddz1[ii,11]=sqrt(.75)

}}

#

twdat=rbind(dmz1,ddz1)

twdat=cbind(twdat,rep(1,(Nmz+Ndz)\*2))

#

colnames(twdat)=c('famnr','twnr','zyg','ph','batch','Ac','A1','A2','Dc','D1','D2','unit')

twdat=as.data.frame(twdat)

twdat$batch=as.factor(twdat$batch)

#

if (wr) write.table(twdat,file='twdat',row.names=F)

#

testr=tryCatch(lme(ph~1,

random=list(

unit=pdIdent(~batch-1),

famnr=pdIdent(~Ac+A1+A2-1),

famnr=pdIdent(~unit-1)

),control=control,data=twdat,method=MLmethod), error = function(e) {-999})

#

testf=tryCatch(lme(ph~1+batch,

random=list(

famnr=pdIdent(~Ac+A1+A2-1),

famnr=pdIdent(~unit-1)

),control=control,data=twdat,method=MLmethod), error = function(e) {-999} )

#

contr=!is.numeric(testr)

contf=!is.numeric(testf)

if (contr) {

nrr=dim(VarCorr(testr))[1]

vcompr=na.omit(matrix(as.numeric((VarCorr(testr))),nrr,2,byrow=F))[c(nbatch,(nbatch+1),(nbatch+4),(nbatch+5)),1]

results[irep,1:4]=vcompr

}

if (contf) {

nrf=dim(VarCorr(testf))[1]

vcompf=na.omit(matrix(as.numeric((VarCorr(testf))),nrf,2,byrow=F))[c(1,4,5),1]

results[irep,5:7]=vcompf

}

#

if (contr) {

b\_eff\_r=testr$coefficients$random$unit

b\_eff\_r=as.numeric(b\_eff\_r)

}

if (contf) {

b\_eff\_f=testf$coefficients$fixed

b\_eff\_f[2:nbatch]=b\_eff\_f[2:nbatch]+b\_eff\_f[1]

b\_eff\_f=as.numeric(b\_eff\_f)

}

#

# plot(beff\_r,beff\_f)

#

if (contf) {

dmz[,2]=keepdmz[,1]-b\_eff\_f[bmz1]

dmz[,3]=keepdmz[,2]-b\_eff\_f[bmz2]

ddz[,2]=keepddz[,1]-b\_eff\_f[bdz1]

ddz[,3]=keepddz[,2]-b\_eff\_f[bdz2]

#

ii=0

for (i in 1:Nmz) {

for (j in 1:2) {

ii=ii+1

dmz1[ii,4]=dmz[i,(1+j)]

}}

#

ii=0

for (i in 1:Ndz) {

for (j in 1:2) {

ii=ii+1

ddz1[ii,4]=ddz[i,(1+j)]

}}

twdat=rbind(dmz1,ddz1)

twdat=cbind(twdat,rep(1,(Nmz+Ndz)\*2))

#

colnames(twdat)=c('famnr','twnr','zyg','ph','batch','Ac','A1','A2','Dc','D1','D2','unit')

twdat=as.data.frame(twdat)

twdat$batch=as.factor(twdat$batch)

#

ctestf=tryCatch(lme(ph~1,

random=list(

famnr=pdIdent(~Ac+A1+A2-1),

famnr=pdIdent(~unit-1)

),control=control,data=twdat,method=MLmethod), error = function(e) {-999}

)

} # end if

#

if (contr) {

dmz[,2]=keepdmz[,1]-b\_eff\_r[bmz1]

dmz[,3]=keepdmz[,2]-b\_eff\_r[bmz2]

ddz[,2]=keepddz[,1]-b\_eff\_r[bdz1]

ddz[,3]=keepddz[,2]-b\_eff\_r[bdz2]

#

ii=0

for (i in 1:Nmz) {

for (j in 1:2) {

ii=ii+1

dmz1[ii,4]=dmz[i,(1+j)]

}}

#

ii=0

for (i in 1:Ndz) {

for (j in 1:2) {

ii=ii+1

ddz1[ii,4]=ddz[i,(1+j)]

}}

#

twdat=rbind(dmz1,ddz1)

twdat=cbind(twdat,rep(1,(Nmz+Ndz)\*2))

#

colnames(twdat)=c('famnr','twnr','zyg','ph','batch','Ac','A1','A2','Dc','D1','D2','unit')

twdat=as.data.frame(twdat)

twdat$batch=as.factor(twdat$batch)

#

ctestr=tryCatch(lme(ph~1,

random=list(

famnr=pdIdent(~Ac+A1+A2-1),

famnr=pdIdent(~unit-1)

),control=control,data=twdat,method=MLmethod), error = function(e) {-999} )

} # end if

#

ccontr=!is.numeric(ctestr)

ccontf=!is.numeric(ctestf)

#

if (ccontr) {

nrr=dim(VarCorr(ctestr))[1]

cvcompr=na.omit(matrix(as.numeric((VarCorr(ctestr))),nrr,2,byrow=F))[c(1,4,5),1]

results[irep,8:10]=cvcompr

}

if (ccontf) {

nrf=dim(VarCorr(ctestf))[1]

cvcompf=na.omit(matrix(as.numeric((VarCorr(ctestf))),nrf,2,byrow=F))[c(1,4,5),1]

results[irep,11:13]=cvcompf

}

#

#

} # irep