Quantitative genetics using the lme4breeding package

Giovanny Covarrubias-Pazaran

2024-05-16

lme4breeding is nice wrapper of the lme4 package that enables the use of especialized plant and animal breeding models that include relationship matrices among individuals (e.g., genomic relationship matrices) and complex covariance structures between factors (e.g., factor analytic structures). It uses all the lme4 machinery for linear and non-linear models, for different response distributions opening a world of possibilities.

The vignettes aim to provide several examples in how to use the lme4breeding package under different scenarios. We will spend the rest of the space providing examples for:

SECTION 1: Basic topics in quantitative genetics

1) Heritability (h^2) calculation
2) Specifying heterogeneous variances in mixed models
3) Half and full diallel designs (using the overlay)
4) Genomic selection (predicting mendelian sampling)
    - GBLUP
    - rrBLUP
5) Indirect genetic effects
6) GCA models and single cross prediction (hybrid prediction)
7) Spatial modeling (using the 2-dimensional splines)
8) Multivariate genetic models and genetic correlations

SECTION 2: Special topics in quantitative genetics

1) Partitioned model
2) UDU' decomposition
3) Mating designs
4) GWAS by GBLUP

SECTION 1: Basic topics in quantitative genetics

1) Marker and non-marker based heritability calculation

Heritability is one of the most popular parameters among the breeding and genetics communities because of the insight it provides in the inheritance of the trait and potential selection response. Heritability is usually estimated as narrow sense (h^2; only additive variance in the numerator σ^2_A), and broad sense (H^2; all genetic variance in the numerator σ^2_G).

In a classical breeding experiment with no molecular markers, special designs are performed to estimate and dissect the additive (σ^2_A) and non-additive (e.g., dominance σ^2_D, and epistatic σ^2_E) variance along with environmental variability. Designs such as generation analysis, North Carolina designs are used to dissect σ^2_A and σ^2_D to estimate the narrow sense heritability (h^2) using only σ^2_A in the numerator. When no special design is available we can still disect the genetic variance (σ^2_G) and estimate the broad sense heritability. In this first example we will show the broad sense estimation which doesn't use covariance matrices for the genotypic effect (e.g., genomic-additive relationship matrices).

The following dataset has 41 potato lines evaluated in 5 locations across 3 years in an RCBD design. We show how to fit the model and extract the variance components to calculate the h^2.

```
library(lme4breeding)
data(DT_example)
DT <- DT_example
A <- A_example

ans1 <- lmebreed(Yield~ (1|Name) + (1|Env) +
                   (1|Env:Name) + (1|Env:Block),
             data=DT)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
vc <- VarCorr(ans1); print(vc,comp=c("Variance"))
```

```
##  Groups    Name        Variance
##  Env:Name  (Intercept)  5.1528
##  Name      (Intercept)  3.7184
##  Env:Block (Intercept)  0.0000
##  Env       (Intercept) 12.0084
##  Residual               4.3661
```

```
ve <- attr(VarCorr(ans1), "sc")^2
n.env <- length(levels(DT$Env))
H2=vc$Name / ( vc$Name + (vc$`Env:Name`/n.env) + (ve/(n.env*2)) )
H2
```

```
##             (Intercept)
## (Intercept)   0.6032732
## attr(,"stddev")
## (Intercept)
##    1.928306
## attr(,"correlation")
##             (Intercept)
## (Intercept)           1
```

That is an estimate of broad-sense heritability.

Recently with markers becoming cheaper, thousand of markers can be run in the breeding materials. When markers are available, a special design is not neccesary to dissect the additive genetic variance. The availability of the additive, dominance and epistatic relationship matrices allow us to estimate σ^2_A, σ^2_D and σ^2_I, although given that A, D and E are not orthogonal the interpretation of models that fit more than the A matrix at the same time becomes cumbersome.

Assume you have a population (even unreplicated) in the field but in addition we have genetic markers. Now we can fit the model and estimate the genomic heritability that explains a portion of the additive genetic variance (with high marker density $\sigma^2_A = \sigma^2_{markers}$)

```
data(DT_cpdata)
DT <- DT_cpdata
GT <- GT_cpdata
MP <- MP_cpdata
#### create the variance-covariance matrix
A <- A.mat(GT) # additive relationship matrix
A <- A + diag(1e-4, ncol(A), ncol(A))
#### look at the data and fit the model
head(DT)
```

```
##         id Row Col Year      color  Yield FruitAver Firmness Rowf Colf
## P003 P003   3   1 2014 0.10075269 154.67     41.93  588.917    3    1
## P004 P004   4   1 2014 0.13891940 186.77     58.79  640.031    4    1
## P005 P005   5   1 2014 0.08681502  80.21     48.16  671.523    5    1
## P006 P006   6   1 2014 0.13408561 202.96     48.24  687.172    6    1
## P007 P007   7   1 2014 0.13519278 174.74     45.83  601.322    7    1
## P008 P008   8   1 2014 0.17406685 194.16     44.63  656.379    8    1
```

```
mix1 <- lmebreed(Yield~ (1|id) + (1|Rowf) + (1|Colf),
                 relmat=list(id=A),
                 control = lmerControl(
                   check.nobs.vs.nlev = "ignore",
                   check.nobs.vs.rankZ = "ignore",
                   check.nobs.vs.nRE="ignore"
                 ),
                 data=DT)
vc <- VarCorr(mix1); print(vc,comp=c("Variance"))
```

```
##  Groups   Name        Variance
##  id       (Intercept)  806.22
##  Colf     (Intercept)  198.43
##  Rowf     (Intercept)  848.57
##  Residual             2992.31
```

```
ve <- attr(VarCorr(mix1), "sc")^2
h2= vc$id / ( vc$id + ve )
as.numeric(h2)
```

```
## [1] 0.2122454
```

In this example we showed how to estimate the additive (σ^2_A) variance components based on markers and estimate narrow-sense heritability (h^2). Notice that we used the `relmat` argument which indicates that the random effect inside the parenthesis (i.e. `id`) has a covariance matrix (A), that will be specified as the Cholesky of the relationship matrix. Please DO NOT provide the inverse to the Cholesky, but rather the original covariance matrix.

2) Specifying heterogeneous variances in univariate models

Very often in multi-environment trials, the assumption that genetic variance is the same across locations may be too naive. Because of that, specifying a general genetic component and a location-specific genetic variance is the way to go.

We estimate variance components for specific environments.

```
data(DT_example)
DT <- DT_example
A <- A_example
head(DT)
```

```
##                     Name     Env Loc Year     Block Yield    Weight
## 33  Manistee(MSL292-A) CA.2013  CA 2013 CA.2013.1     4 -1.904711
## 65          CO02024-9W CA.2013  CA 2013 CA.2013.1     5 -1.446958
## 66  Manistee(MSL292-A) CA.2013  CA 2013 CA.2013.2     5 -1.516271
## 67            MSL007-B CA.2011  CA 2011 CA.2011.2     5 -1.435510
## 68           MSR169-8Y CA.2013  CA 2013 CA.2013.1     5 -1.469051
## 103         AC05153-1W CA.2013  CA 2013 CA.2013.1     6 -1.307167
## Compound simmetry (CS) + Diagonal (DIAG) model
Z <- with(DT, dsc(Env))$Z
csdiagFormula <- paste0( "Yield ~ Env + (", paste(colnames(Z), collapse = "+"), "|| Name)")
for(i in 1:ncol(Z)){DT[,colnames(Z)[i]] <- Z[,i]}
ansCSDG <- lmebreed(as.formula(csdiagFormula),
                    relmat = list(Name = A ),
                    data=DT)
vc <- VarCorr(ansCSDG); print(vc,comp=c("Variance"))
```

```
##  Groups   Name        Variance
##  Name     (Intercept)  2.9638
##  Name.1   CA.2011     10.4259
##  Name.2   CA.2012      2.6589
##  Name.3   CA.2013      5.7021
##  Residual              4.3976
```

In the previous example we showed how the left side of the formulae to specify the covariance structure and the effect on the right side (`covStructure | effect`) in the `lmebreed()` solver. By specifying (`CA.2011+CA.2012+CA.2013 || Name`) we declare a covariance structure for Name between environments. The `||` indicates that we do not want to fit covariance between these environments This is considered a CS + DIAG (compound symmetry + diagonal) model.

3) Half and full diallel designs (use of the overlay)

When breeders are looking for the best single-cross combinations, diallel designs have been by far the most used design in crops like maize. There are 4 types of diallel designs depending on whether reciprocal and self-crosses (omission of parents) are performed (full diallel with parents n^2; full diallel without parents n(n-1); half diallel with parents 1/2 * n(n+1); half diallel without parents 1/2 * n(n-1)). In this example we will show a full diallel design (reciprocal crosses are performed) and half diallel designs (only one of the directions is performed).

In the first data set we show a full diallel among 40 lines from 2 heterotic groups, 20 in each. Therefore 400 possible hybrids are possible. We have pehnotypic data for 100 of them across 4 locations. We use the data available to fit a model of the form:

$y = X\beta + Zu_1 + Zu_2 + Zu_S + \epsilon$

We estimate variance components for GCA_1, GCA_2 and SCA and use them to estimate heritability. Additionally BLUPs for GCA and SCA effects can be used to predict crosses.

```
data(DT_cornhybrids)
DT <- DT_cornhybrids
DTi <- DTi_cornhybrids
GT <- GT_cornhybrids

modFD <- lmebreed(Yield~Location + (1|GCA1)+(1|GCA2)+(1|SCA),
              data=DT)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
vc <- VarCorr(modFD); print(vc,comp=c("Variance"))
```

```
##  Groups   Name        Variance
##  SCA      (Intercept) 187.6620
##  GCA2     (Intercept)   7.2901
##  GCA1     (Intercept)   0.0000
##  Residual             221.1425
```

```r
Vgca <- vc$GCA1 + vc$GCA2
Vsca <- vc$SCA
Ve <- attr(vc, "sc")^2
Va = 4*Vgca
Vd = 4*Vsca
Vg <- Va + Vd
(H2 <- Vg / (Vg + (Ve)) )
```

```
##             (Intercept)
## (Intercept)   0.7790676
## attr(,"stddev")
## (Intercept)
##           0
## attr(,"correlation")
##             (Intercept)
## (Intercept)           1
```

```r
(h2 <- Va / (Vg + (Ve)) )
```

```
##             (Intercept)
## (Intercept)  0.02913284
## attr(,"stddev")
## (Intercept)
##           0
## attr(,"correlation")
##             (Intercept)
## (Intercept)           1
```

Don't worry too much about the `h2` value, the data was simulated to be mainly dominance variance, therefore the `Va` was simulated extremely small leading to such value of narrow sense `h2`.

In the second data set we show a small half diallel with 7 parents crossed in one direction. There are n(n-1)/2 possible crosses; 7(6)/2 = 21 unique crosses. Parents appear as males or females indistictly. Each with two replications in a CRD. For a half diallel design a single GCA variance component for both males and females can be estimated and an SCA as well ($\sigma^2_G CA$ and $\sigma^2_S CA$ respectively), and BLUPs for GCA and SCA of the parents can be extracted. We will show how to do so using the `overlay()` function. The specific model here is:

$y = X\beta + Zu_g + Zu_s + \epsilon$

```r
data("DT_halfdiallel")
DT <- DT_halfdiallel
head(DT)
```

```
##   rep geno male female     sugar
## 1   1   12    1      2 13.950509
## 2   2   12    1      2  9.756918
## 3   1   13    1      3 13.906355
## 4   2   13    1      3  9.119455
## 5   1   14    1      4  5.174483
## 6   2   14    1      4  8.452221
```

```r
DT$femalef <- as.factor(DT$female)
DT$malef <- as.factor(DT$male)
DT$genof <- as.factor(DT$geno)
# overlay matrix to be added to the addmat argument
Z <- with(DT, overlay(femalef,malef) )
```

```r
Z <- Z[which(!is.na(DT$sugar)),]
# create inital values for incidence matrix but irrelevant
# since these will be replaced by admat argument
fema <- (rep(colnames(Z), nrow(DT)))[1:nrow(DT)]
#### model using overlay without relationship matrix
modh <- lmebreed(sugar ~ (1|genof) + (1|fema),
                 addmat = list(fema=Z),
             data=DT)
```

```
## boundary (singular) fit: see help('isSingular')
```

```r
vc <- VarCorr(modh); print(vc,comp=c("Variance"))
```

```
##  Groups   Name        Variance
##  genof    (Intercept) 1.8160
##  fema     (Intercept) 5.5088
##  Residual             3.1174
```

```r
ve <- attr(vc, "sc")^2;ve
```

```
## [1] 3.117351
```

Notice how the `overlay()` argument makes the overlap of incidence matrices possible making sure that male and female are joint into a single random effect.

4) Genomic selection: predicting mendelian sampling

In this section we will use wheat data from CIMMYT to show how genomic selection is performed. This is the case of prediction of specific individuals within a population. It basically uses a similar model of the form:

$y = X\beta + Zu + \epsilon$

and takes advantage of the variance covariance matrix for the genotype effect known as the additive relationship matrix (A) and calculated using the `A.mat` function to establish connections among all individuals and predict the BLUPs for individuals that were not measured. The prediction accuracy depends on several factors such as the heritability (h^2), training population used (TP), size of TP, etc.

```r
# data(DT_wheat)
# DT <- DT_wheat
# GT <- GT_wheat[,1:200]
# colnames(DT) <- paste0("X",1:ncol(DT))
# DT <- as.data.frame(DT);DT$line <- as.factor(rownames(DT))
# # select environment 1
# rownames(GT) <- rownames(DT)
# K <- A.mat(GT) # additive relationship matrix
# colnames(K) <- rownames(K) <- rownames(DT)
# # GBLUP pedigree-based approach
# set.seed(12345)
# y.trn <- DT
# vv <- sample(rownames(DT),round(nrow(DT)/5))
# y.trn[vv,"X1"] <- NA
# head(y.trn)
# ## GBLUP
# K <- K + diag(1e-4, ncol(K), ncol(K) )
# ans <- lmebreed(X1 ~ (1|line),
#                  relmat = list(line=K),
#                  control = lmerControl(
#                    check.nobs.vs.nlev = "ignore",
#                    check.nobs.vs.rankZ = "ignore",
#                    check.nobs.vs.nRE="ignore"
#                  ),
#                  data=y.trn)
# vc <- VarCorr(ans); print(vc,comp=c("Variance"))
#
# # take a extended dataset and fit a dummy model
# # just to get required matrices
# y.tst <- y.trn; y.tst$X1 <- imputev(y.tst$X1)
# ans2 <- update(ans,
#                 start = getME(ans, "theta"),
#                 data = y.tst,
#                 control = lmerControl(check.nobs.vs.nlev = "ignore",
#                                       check.nobs.vs.rankZ = "ignore",
#                                       check.nobs.vs.nRE="ignore",
#                                       optCtrl = list(maxeval= 1),
#                                       calc.derivs = FALSE))
# # compute predictive ability
# cor(ranef(ans2)$line[vv,],DT[vv,"X1"], use="complete")
# # # other approach
# # mme <- getMME(ans2, vc=vc, recordsToKeep = which(!is.na(y.trn$X1)))
# # cor(mme$bu[vv,],DT[vv,"X1"], use="complete")
#
# ## rrBLUP
# M <- tcrossprod(GT)
# xx <- with(y.trn, redmm(x=line, M=M, nPC=100, returnLam = TRUE))
# Z <- xx$Z[which(!is.na(y.trn$X1)),]
# custom <- (rep(colnames(Z), nrow(DT)))[1:nrow(DT)]
# ansRRBLUP <- lmebreed(X1 ~ (1|custom),
#                       addmat = list(custom=Z),
#                       data=y.trn)
# re <- ranef(ansRRBLUP)$custom
# u = tcrossprod(xx$Lam, t(as.matrix( re[colnames(xx$Lam),] ) ))
# cor(u[vv,],DT[vv,"X1"], use="complete")
```

5) Indirect genetic effects

General variance structures can be used to fit indirect genetic effects. Here, we use an example dataset to show how we can fit the variance and covariance components between two or more different random effects. We now fit the indirect genetic effects model with covariance between DGE and IGE. On top of that we can include a relationship matrix for the two random effects that are being forced to co-vary

```
# data(DT_ige)
# DT <- DT_ige
# A_ige <- A_ige + diag(1e-4, ncol(A_ige), ncol(A_ige) )
# # Define 2 dummy variables to make a fake covariance
# # for two different random effects
# DT$fn <- DT$nn <- 1
# # Create the incidence matrix for the first random effect
# Zf <- Matrix::sparse.model.matrix( ~ focal-1, data=DT )
# colnames(Zf) <- gsub("focal","", colnames(Zf))
# Zf <- Zf[which(!is.na(DT$trait)),]
# # Create the incidence matrix for the second random effect
# Zn <- Matrix::sparse.model.matrix( ~ neighbour-1, data=DT )
# colnames(Zn) <- gsub("neighbour","", colnames(Zn))
# Zn <- Zn[which(!is.na(DT$trait)),]
# # Make inital values for incidence matrix but irrelevant
# # since these will be replaced by the addmat argument
# both <- (rep(colnames(Zf), nrow(DT)))[1:nrow(DT)]
# # Fit the model
# modIGE <- lmebreed(trait ~ block + (0+fn+nn|both),
#                    addmat = list(both=list(Zf,Zn)),
#                    relmat = list(both=A_ige),
#                    data = DT)
# vc <- VarCorr(modIGE); print(vc,comp=c("Variance"))
# blups <- ranef(modIGE)
# pairs(blups$both)
# cov2cor(vc$both)
```

6) Genomic selection: single cross prediction

When doing prediction of single cross performance the phenotype can be dissected in three main components, the general combining abilities (GCA) and specific combining abilities (SCA). This can be expressed with the same model analyzed in the diallel experiment mentioned before:

$y = X\beta + Zu_1 + Zu_2 + Zu_S + \epsilon$

with:

u_1 ~ N(0, $K_1\sigma^2_u 1$)

u_2 ~ N(0, $K_2\sigma^2_u 2$)

u_s ~ N(0, $K_3\sigma^2_u s$)

And we can specify the K matrices. The main difference between this model and the full and half diallel designs is the fact that this model will include variance covariance structures in each of the three random effects (GCA1, GCA2 and SCA) to be able to predict the crosses that have not ocurred yet. We will use the data published by Technow et al. (2015) to show how to do prediction of single crosses.

```
# data(DT_technow)
# DT <- DT_technow
# Md <- (Md_technow*2) - 1
# Mf <- (Mf_technow*2) - 1
# Ad <- A.mat(Md)
# Af <- A.mat(Mf)
# Ad <- Ad + diag(1e-4, ncol(Ad), ncol(Ad))
# Af <- Af + diag(1e-4, ncol(Af), ncol(Af))
# # simulate some missing hybrids to predict
# y.trn <- DT
# vv1 <- which(!is.na(DT$GY))
# vv2 <- sample(DT[vv1,"hy"], 100)
# y.trn[which(y.trn$hy %in% vv2),"GY"] <- NA
# ans2 <- lmebreed(GY ~ (1|dent) + (1|flint),
#                  relmat = list(dent=Ad,
#                                flint=Af),
#                  data=y.trn)
# vc <- VarCorr(ans2); print(vc,comp=c("Variance"))
```

```
#
# # take a extended dataset and fit a dummy model
# # just to get required matrices
# y.tst <- y.trn; y.tst$GY <- imputev(y.tst$GY)
# ans2p <- update(ans2,
#                  start = getME(ans2, "theta"),
#                  data = y.tst,
#                  control = lmerControl(check.nobs.vs.nlev = "ignore",
#                                        check.nobs.vs.rankZ = "ignore",
#                                        check.nobs.vs.nRE="ignore",
#                                        optCtrl = list(maxeval= 1),
#                                        calc.derivs = FALSE))
#
# re <- ranef(ans2p)
#
# Pdent <- as.matrix(re$dent[,1,drop=FALSE]) %*% Matrix(1, ncol=nrow(re$flint), nrow=1)
# Pflint <- as.matrix(re$flint[,1,drop=FALSE]) %*% Matrix(1, ncol=nrow(re$dent), nrow=1)
# P <- Pdent + t(Pflint); colnames(P) <- rownames(re$flint)
#
# preds <- real <- numeric()
# for(iHyb in vv2){
#   parents <- strsplit(iHyb,":")[[1]]
#   preds[iHyb] <- P[which(rownames(P) %in% parents),which(colnames(P) %in% parents)]
#   real[iHyb] <- DT[which(DT$hy == iHyb),"GY"]
# }
# plot(preds, real)
# cor(preds, real)
```

In the previous model we only used the GCA effects (GCA1 and GCA2) for practicity, altough it's been shown that the SCA effect doesn't actually help that much in increasing prediction accuracy, but does increase a lot the computation intensity required since the variance covariance matrix for SCA is the kronecker product of the variance covariance matrices for the GCA effects, resulting in a 10578 x 10578 matrix that increases in a very intensive manner the computation required.

A model without covariance structures would show that the SCA variance component is insignificant compared to the GCA effects. This is why including the third random effect doesn't increase the prediction accuracy.

8) Spatial modeling: using the 2-dimensional spline

We will use the CPdata to show the use of 2-dimensional splines for accomodating spatial effects in field experiments. In early generation variety trials the availability of seed is low, which makes the use of unreplicated designs a neccesity more than anything else. Experimental designs such as augmented designs and partially-replicated (p-rep) designs are becoming ever more common these days.

In order to do a good job modeling the spatial trends happening in the field, special covariance structures have been proposed to accomodate such spatial trends (i.e. autoregressive residuals; ar1). Unfortunately, some of these covariance structures make the modeling rather unstable. More recently, other research groups have proposed the use of 2-dimensional splines to overcome such issues and have a more robust modeling of the spatial terms (Lee et al. 2013; Rodríguez-Álvarez et al. 2018).

In this example we assume an unreplicated population where row and range information is available which allows us to fit a 2 dimensional spline model.

```
data(DT_cpdata)
DT <- DT_cpdata
```

```
# add the units column
DT$units <- as.factor(1:nrow(DT))
# get spatial incidence matrix
Zs <- with(DT, tps(Row, Col))$All
rownames(Zs) <- DT$units
# reduce the matrix to its PCs
Z = with(DT, redmm(x=units, M=Zs, nPC=100))
# make sure dimensions will be correct
Z <- Z[which(!is.na(DT$Yield)),]
# create dummy variable
spatial <- (rep(colnames(Z), nrow(DT)))[1:nrow(DT)]
# fit model
mix1 <- lmebreed(Yield~ (1|Rowf) + (1|Colf) + (1|spatial),
                 addmat =list(spatial=Z),
                 control = lmerControl(
                   check.nobs.vs.nlev = "ignore",
                   check.nobs.vs.rankZ = "ignore",
                   check.nobs.vs.nRE="ignore"
                 ),
                 data=DT)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
vc <- VarCorr(mix1); print(vc,comp=c("Variance"))
```

```
##  Groups   Name        Variance
##  spatial  (Intercept)  446.67
##  Colf     (Intercept)  157.84
##  Rowf     (Intercept)  819.15
##  Residual             3533.82
```

Notice that the job is done by the `spl2Da()` function that takes the `Row` and `Col` information to fit a spatial kernel.

9) Multivariate genetic models and genetic correlations

Sometimes is important to estimate genetic variance-covariance among traits–multi-reponse models are very useful for such a task. Let see an example with 2 traits (`color`, `Yield`) and a single random effect (genotype; `id`) although multiple effects can be modeled as well. We need to use a variance covariance structure for the random effect to be able to obtain the genetic covariance among traits.

```
# data(DT_cpdata)
# DT <- DT_cpdata
# GT <- GT_cpdata
# MP <- MP_cpdata
# #### create the variance-covariance matrix
# A <- A.mat(GT) # additive relationship matrix
# A <- A + diag(1e-4, ncol(A), ncol(A))
# #### look at the data and fit the model
# head(DT)
# DT2 <- stackTrait(data=DT, traits = c("Yield","color"))
# head(DT2$long)
#
# mix1 <- lmebreed(valueS~ (0+trait|id),
#                    relmat=list(id=A),
#                   control = lmerControl(
#                     check.nobs.vs.nlev = "ignore",
#                     check.nobs.vs.rankZ = "ignore",
#                     check.nobs.vs.nRE="ignore"
#                   ),
#                   data=DT2$long)
# vc <- VarCorr(mix1); print(vc,comp=c("Variance"))
```

Now you can extract the BLUPs using `ranef(ans.m)`. Also, genetic correlations and heritabilities can be calculated easily.

```
# cov2cor(vc$id)
```

SECTION 2: Special topics in Quantitative genetics

1) Partitioned model

The partitioned model was popularized by () to show that marker effects can be obtained by fitting a GBLUP model to reduce the computational burden and then recover them by creating some special matrices MM' for GBLUP and M'(M'M)- to recover marker effects. Here we show a very easy example using the DT_cpdata:

```
# data("DT_cpdata")
# DT <- as.data.frame(DT_cpdata)
# M <- GT_cpdata
#
# ################
# # PARTITIONED GBLUP MODEL
# ################
#
# MMT <-tcrossprod(M) ## MM' = additive relationship matrix
# MMTinv<-solve(MMT) ## inverse
# MTMMTinv<-t(M)%*%MMTinv # M' %*% (M'M)-
#
# mix.part <- lmebreed(color ~ (1|id),
#                       relmat = list(id=MMT),
#                       control = lmerControl(
#                         check.nobs.vs.nlev = "ignore",
#                         check.nobs.vs.rankZ = "ignore",
#                         check.nobs.vs.nRE="ignore"
#                       ),
#                       data=DT)
#
# #convert BLUPs to marker effects me=M'(M'M)- u
# re <- ranef(mix.part)$id
# me.part<-MTMMTinv[,rownames(re)]%*%matrix(re[,1],ncol=1)
# plot(me.part)
```

As can be seen, these two models are equivalent with the exception that the partitioned model is more computationally efficient.

2) UDU' decomposition

Lee and Van der Warf (2015) proposed a decomposition of the relationship matrix A=UDU' together with a transformation of the response and fixed effects Uy = Ux + UZ + e, to fit a model where the phenotypic variance matrix V is a diagonal because the relationship matrix is the diagonal matrix D from the decomposition that can be inverted easily and make multitrait models more feasible.

```
#
# data("DT_wheat")
# rownames(GT_wheat) <- rownames(DT_wheat)
# G <- A.mat(GT_wheat)
# Y <- data.frame(DT_wheat)
#
# # make the decomposition
# UD<-eigen(G) # get the decomposition: G = UDU'
# U<-UD$vectors
# D<-diag(UD$values)# This will be our new 'relationship-matrix'
# rownames(D) <- colnames(D) <- rownames(G)
# X<-model.matrix(~1, data=Y) # here: only one fixed effect (intercept)
# UX<-t(U)%*%X # premultiply X and y by U'
# UY <- t(U) %*% as.matrix(Y) # multivariate
#
# # dataset for decomposed model
# DTd<-data.frame(id = rownames(G) ,UY, UX =UX[,1])
# DTd$id<-as.character(DTd$id)
# head(DTd)
#
# modeld <- lmebreed(X1~ UX + (1|id),
#                  relmat=list(id=D),
#                  control = lmerControl(
#                    check.nobs.vs.nlev = "ignore",
#                    check.nobs.vs.rankZ = "ignore",
#                    check.nobs.vs.nRE="ignore"
#                  ),
#                  data=DTd)
# vc <- VarCorr(modeld); print(vc,comp=c("Variance"))
#
# # dataset for normal model
# DTn<-data.frame(id = rownames(G) , DT_wheat)
# DTn$id<-as.character(DTn$id)
#
# modeln <- lmebreed(X1~ (1|id),
#                    relmat=list(id=G),
#                    control = lmerControl(
#                      check.nobs.vs.nlev = "ignore",
#                      check.nobs.vs.rankZ = "ignore",
#                      check.nobs.vs.nRE="ignore"
#                    ),
#                    data=DTn)
#
# ## compare regular and transformed blups
# red <- ranef(modeld)$id
# ren <- ranef(modeln)$id
# plot(x=(solve(t(U)))%*%  red[colnames(D),],
```

```r
#       y=ren[colnames(D),],
#       xlab="UDU blup", ylab="blup")
#
```

As can be seen, the two models are equivalent. Despite the fact that lme4breeding doesn't take a great advantage of this trick because it was built for dense matrices using the Armadillo library. Other software may be better using this trick.

3) Mating designs

Estimating variance components has been a topic of interest for the breeding community for a long time. Here we show how to calculate additive and dominance variance using the North Carolina Design I (Nested design) and North Carolina Design II (Factorial design) using the classical Expected Mean Squares method and the REML methods from lme4breeding and how these two are equivalent.

```r
data(DT_expdesigns)
DT <- DT_expdesigns$car1
DT <- aggregate(yield~set+male+female+rep, data=DT, FUN = mean)
DT$setf <- as.factor(DT$set)
DT$repf <- as.factor(DT$rep)
DT$malef <- as.factor(DT$male)
DT$femalef <- as.factor(DT$female)
#levelplot(yield~male*female|set, data=DT, main="NC design I")
##############################
## Expected Mean Square method
##############################
mix1 <- lm(yield~ setf + setf:repf + femalef:malef:setf + malef:setf, data=DT)
MS <- anova(mix1); MS
```

North Carolina Design I (Nested design)

```
## Analysis of Variance Table
##
## Response: yield
##                     Df Sum Sq Mean Sq F value   Pr(>F)
## setf                 1 0.1780 0.17796  1.6646 0.226012
## setf:repf            2 0.9965 0.49824  4.6605 0.037141 *
## setf:malef           4 7.3904 1.84759 17.2822 0.000173 ***
## setf:femalef:malef   6 1.6083 0.26806  2.5074 0.095575 .
## Residuals           10 1.0691 0.10691
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
ms1 <- MS["setf:malef","Mean Sq"]
ms2 <- MS["setf:femalef:malef","Mean Sq"]
mse <- MS["Residuals","Mean Sq"]
nrep=2
nfem=2
Vfm <- (ms2-mse)/nrep
Vm <- (ms1-ms2)/(nrep*nfem)

## Calculate Va and Vd
Va=4*Vm # assuming no inbreeding (4/(1+F))
```

```r
Vd=4*(Vfm-Vm) # assuming no inbreeding(4/(1+F)^2)
Vg=c(Va,Vd); names(Vg) <- c("Va","Vd"); Vg
```

```
##        Va        Vd
##  1.579537 -1.257241
```

```r
###############################
## REML method
###############################
mix2 <- lmebreed(yield~ setf + setf:repf +
                   (1|femalef:malef:setf) + (1|malef:setf),
             data=DT)
vc <- VarCorr(mix2); print(vc,comp=c("Variance"))
```

```
##  Groups             Name        Variance
##  femalef:malef:setf (Intercept) 0.080574
##  malef:setf         (Intercept) 0.394884
##  Residual                       0.106907
```

```r
Vfm <- vc$`femalef:malef:setf`
Vm <- vc$`malef:setf`

## Calculate Va and Vd
Va=4*Vm # assuming no inbreeding (4/(1+F))
Vd=4*(Vfm-Vm) # assuming no inbreeding(4/(1+F)^2)
Vg=c(Va,Vd); names(Vg) <- c("Va","Vd"); Vg
```

```
##        Va        Vd
##  1.579537 -1.257240
```

As can be seen the REML method is easier than manipulating the MS and we arrive to the same results.

```r

DT <- DT_expdesigns$car2
DT <- aggregate(yield~set+male+female+rep, data=DT, FUN = mean)
DT$setf <- as.factor(DT$set)
DT$repf <- as.factor(DT$rep)
DT$malef <- as.factor(DT$male)
DT$femalef <- as.factor(DT$female)
#levelplot(yield~male*female|set, data=DT, main="NC desing II")
head(DT)
```

North Carolina Design II (Factorial design)

```
##   set male female rep   yield setf repf malef femalef
## 1   1    1      1   1  831.03    1    1     1       1
## 2   1    2      1   1 1046.55    1    1     2       1
## 3   1    3      1   1  853.33    1    1     3       1
## 4   1    4      1   1  940.00    1    1     4       1
## 5   1    5      1   1  802.00    1    1     5       1
## 6   1    1      2   1  625.93    1    1     1       2
```

```r
N=with(DT,table(female, male, set))
nmale=length(which(N[1,,1] > 0))
nfemale=length(which(N[,1,1] > 0))
nrep=table(N[,,1])
```

```r
nrep=as.numeric(names(nrep[which(names(nrep) !=0)]))

###############################
## Expected Mean Square method
###############################

mix1 <- lm(yield~ setf + setf:repf +
             femalef:malef:setf + malef:setf +
             femalef:setf,
           data=DT)
MS <- anova(mix1); MS
```

```
## Analysis of Variance Table
##
## Response: yield
##                     Df  Sum Sq Mean Sq F value    Pr(>F)
## setf                 1  847836  847836 45.6296 1.097e-09 ***
## setf:repf            4  144345   36086  1.9421  0.109652
## setf:malef           8  861053  107632  5.7926 5.032e-06 ***
## setf:femalef         8  527023   65878  3.5455  0.001227 **
## setf:femalef:malef  32  807267   25227  1.3577  0.129527
## Residuals           96 1783762   18581
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
ms1 <- MS["setf:malef","Mean Sq"]
ms2 <- MS["setf:femalef","Mean Sq"]
ms3 <- MS["setf:femalef:malef","Mean Sq"]
mse <- MS["Residuals","Mean Sq"]
nrep=length(unique(DT$rep))
nfem=length(unique(DT$female))
nmal=length(unique(DT$male))
Vfm <- (ms3-mse)/nrep;
Vf <- (ms2-ms3)/(nrep*nmale);
Vm <- (ms1-ms3)/(nrep*nfemale);

Va=4*Vm; # assuming no inbreeding (4/(1+F))
Va=4*Vf; # assuming no inbreeding (4/(1+F))
Vd=4*(Vfm); # assuming no inbreeding(4/(1+F)^2)
Vg=c(Va,Vd); names(Vg) <- c("Va","Vd"); Vg
```

```
##        Va        Vd
## 10840.192  8861.659
```

```r
###############################
## REML method
###############################

mix2 <- lmebreed(yield~ setf + setf:repf +
               (1|femalef:malef:setf) + (1|malef:setf) +
               (1|femalef:setf),
             data=DT)
vc <- VarCorr(mix2); print(vc,comp=c("Variance"))
```

```
##  Groups             Name        Variance
##  femalef:malef:setf (Intercept)  2215.4
##  malef:setf         (Intercept)  5493.6
##  femalef:setf       (Intercept)  2710.0
##  Residual                       18580.9
```

```
Vfm <- vc$`femalef:malef:setf`
Vm <- vc$`malef:setf`
Vf <- vc$`femalef:setf`

Va=4*Vm; # assuming no inbreeding (4/(1+F))
Va=4*Vf; # assuming no inbreeding (4/(1+F))
Vd=4*(Vfm); # assuming no inbreeding(4/(1+F)^2)
Vg=c(Va,Vd); names(Vg) <- c("Va","Vd"); Vg
```

```
##        Va        Vd
## 10840.199  8861.682
```

As can be seen, the REML method is easier than manipulating the MS and we arrive to the same results.

4) GWAS by GBLUP

Gualdron-Duarte et al. (2014) and Bernal-Rubio et al. (2016) proved that in (SingleStep)GBLUP or RRBLUP/SNP-BLUP, dividing the estimate of the marker effect by its standard error is mathematically equivalent to fixed regression EMMAX GWAS, even if markers are estimated as random effects in GBLUP and as fixed effects in EMMAX. That way fitting a GBLUP model is enough to perform GWAS for additive and on-additive effects.

Let us use the DT_cpdata dataset to explore the GWAS by GBLUP method

```
data(DT_cpdata)
DT <- DT_cpdata
GT <- GT_cpdata#[,1:200]
MP <- MP_cpdata
M<- GT
n <- nrow(DT) # to be used for degrees of freedom
k <- 1 # to be used for degrees of freedom (number of levels in fixed effects)
```

Instead of fitting the RRBLUP/SNP-BLUP model we can fit a GBLUP model which is less computationally demanding and recover marker effects and their standard errors from the genotype effects.

```
# ###########################
# #### GWAS by GBLUP approach
# ###########################
# MMT <-tcrossprod(M) ## MM' = additive relationship matrix
# MMT <- MMT + diag(1e-4, ncol(MMT), ncol(MMT) )
# MMTinv<-solve( MMT ) ## inverse
# MTMMTinv<-t(M)%*%MMTinv # M' %*% (M'M)-
#
# mix.part <- lmebreed(color ~ (1|id) + (1|Rowf) + (1|Colf),
#                        relmat = list(id=MMT),
#                        control = lmerControl(
#                          check.nobs.vs.nlev = "ignore",
#                          check.nobs.vs.rankZ = "ignore",
#                          check.nobs.vs.nRE="ignore"
#                        ),
#                        data=DT)
```

```r
# vc <- VarCorr(mix.part); print(vc,comp=c("Variance"))
# mme <- getMME(object=mix.part)
# #convert BLUPs to marker effects me=M'(M'M)- u
# re <- ranef(mix.part)$id
# a.from.g<-MTMMTinv[,rownames(re)]%*%matrix(re[,1],ncol=1)
# var.g <- kronecker(MMT[rownames(re),rownames(re)],vc$id) -
#   mme$Ci[rownames(re),rownames(re) ]
# var.a.from.g <- t(M)%*%MMTinv[,rownames(re)]%*% (var.g) %*% t(MMTinv[,rownames(re)])%*%M
# se.a.from.g <- sqrt(diag(var.a.from.g))
# t.stat.from.g <- a.from.g/se.a.from.g # t-statistic
# pvalGBLUP <- dt(t.stat.from.g,df=n-k-1) # -log10(pval)
```

Now we can look at the p-values coming from the 3 approaches to indeed show that results are equivalent.

```r
# plot(-log(pvalGBLUP), main="GWAS by GBLUP")
```

Literature

Giovanny Covarrubias-Pazaran (2024). lme4breeding: enabling genetic evaluation in the age of genomic data. To be submitted to Bioinformatics.

Bates Douglas, Maechler Martin, Bolker Ben, Walker Steve. 2015. Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software, 67(1), 1-48.

Bernardo Rex. 2010. Breeding for quantitative traits in plants. Second edition. Stemma Press. 390 pp.

Gilmour et al. 1995. Average Information REML: An efficient algorithm for variance parameter estimation in linear mixed models. Biometrics 51(4):1440-1450.

Henderson C.R. 1975. Best Linear Unbiased Estimation and Prediction under a Selection Model. Biometrics vol. 31(2):423-447.

Kang et al. 2008. Efficient control of population structure in model organism association mapping. Genetics 178:1709-1723.

Lee, D.-J., Durban, M., and Eilers, P.H.C. (2013). Efficient two-dimensional smoothing with P-spline ANOVA mixed models and nested bases. Computational Statistics and Data Analysis, 61, 22 - 37.

Lee et al. 2015. MTG2: An efficient algorithm for multivariate linear mixed model analysis based on genomic information. Cold Spring Harbor. doi: http://dx.doi.org/10.1101/027201.

Maier et al. 2015. Joint analysis of psychiatric disorders increases accuracy of risk prediction for schizophrenia, bipolar disorder, and major depressive disorder. Am J Hum Genet; 96(2):283-294.

Rodriguez-Alvarez, Maria Xose, et al. Correcting for spatial heterogeneity in plant breeding experiments with P-splines. Spatial Statistics 23 (2018): 52-71.

Searle. 1993. Applying the EM algorithm to calculating ML and REML estimates of variance components. Paper invited for the 1993 American Statistical Association Meeting, San Francisco.

Yu et al. 2006. A unified mixed-model method for association mapping that accounts for multiple levels of relatedness. Genetics 38:203-208.

Tunnicliffe W. 1989. On the use of marginal likelihood in time series model estimation. JRSS 51(1):15-27.