

Quantitative genetics using the sommer package

Giovanny Covarrubias-Pazaran

2020-06-25

The sommer package was developed to provide R users a powerful and reliable multivariate mixed model solver for different genetic and non-genetic analysis in diploid and polyploid organisms. This package allows the user to estimate variance components for a mixed model with the advantage of specifying the variance-covariance structure of the random effects, specify heterogeneous variances, and obtain other parameters such as BLUPs, BLUEs, residuals, fitted values, variances for fixed and random effects, etc. The core algorithms of the package are coded in C++ using the Armadillo library to optimize dense matrix operations common in the direct-inversion algorithms.

The package is focused on problems of the type $p > n$ related to genomic prediction (hybrid prediction & genomic selection) and GWAS analysis, although any general mixed model can be fitted as well. The package provides kernels to estimate additive (**A.mat**), dominance (**D.mat**), and epistatic (**E.mat**) relationship matrices that have been shown to increase prediction accuracy under certain scenarios or simply to estimate the variance components of such. The package provides flexibility to fit other genetic models such as full and half diallel models as well.

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

SECTION 1: Introduction

- 1) Background in linear algebra

SECTION 2: Topics in quantitative genetics

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

SECTION 3: Special topics in quantitative genetics

- 1) Partitioned model
- 2) UDU' decomposition
- 3) Mating designs
- 4) Dominance variance

SECTION 1: Introduction

Backgrounds in linear algebra

The core of the package the `mmer` function which solve the mixed model equations. The functions are an interface to call the NR Direct-Inversion Newton-Raphson or Average Information (Tunnickliffe 1989; Gilmour et al. 1995; Lee et al. 2016). Since version 2.0 sommer can handle multivariate models. Following Maier et al. (2015), the multivariate (and by extension the univariate) mixed model implemented has the form:

$$y_1 = X_1\beta_1 + Z_1u_1 + \epsilon_1 \quad y_2 = X_2\beta_2 + Z_2u_2 + \epsilon_2 \quad \dots \quad y_i = X_i\beta_i + Z_iu_i + \epsilon_i$$

where y_i is a vector of trait phenotypes, β_i is a vector of fixed effects, u_i is a vector of random effects for individuals and e_i are residuals for trait 'i' ($i = 1, \dots, t$). The random effects ($u_1 \dots u_t$ and e_i) are assumed to be normally distributed with mean zero. X and Z are incidence matrices for fixed and random effects respectively. The distribution of the multivariate response and the phenotypic variance covariance (V) are:

$$Y = X\beta + ZU + \epsilon_i$$

$$Y \sim \text{MVN}(X\beta, V)$$

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_t \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} X_1 & \dots & \dots \\ \vdots & \ddots & \vdots \\ \dots & \dots & X_t \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} Z_1K\sigma_{g_1}^2Z_1' + H\sigma_{\epsilon_1}^2 & \dots & Z_1K\sigma_{g_{1,t}}Z_t' + H\sigma_{\epsilon_{1,t}} \\ \vdots & \ddots & \vdots \\ Z_1K\sigma_{g_{1,t}}Z_t' + H\sigma_{\epsilon_{1,t}} & \dots & Z_tK\sigma_{g_t}^2Z_t' + H\sigma_{\epsilon_t}^2 \end{bmatrix}$$

where K is the relationship or covariance matrix for the k th random effect ($u=1, \dots, k$), and $R=I$ is an identity matrix for the residual term. The terms $\sigma_{g_i}^2$ and $\sigma_{\epsilon_i}^2$ denote the genetic (or any of the k th random terms) and residual variance of trait 'i', respectively and $\sigma_{g_{ij}}$ and $\sigma_{\epsilon_{ij}}$ the genetic (or any of the k th random terms) and residual covariance between traits 'i' and 'j' ($i=1, \dots, t$, and $j=1, \dots, t$). The algorithm implemented optimizes the log likelihood:

$$\log L = 1/2 * \ln(|V|) + \ln(X'V|X) + Y'PY$$

where $| |$ is the determinant of a matrix. And the REML estimates are updated using a Newton optimization algorithm of the form:

$$\theta^{k+1} = \theta^k + (H^k)^{-1} * \frac{dL}{d\sigma_i^2} | \theta^k$$

Where, θ is the vector of variance components for random effects and covariance components among traits, H^{-1} is the inverse of the Hessian matrix of second derivatives for the k th cycle, $\frac{dL}{d\sigma_i^2}$ is the vector of first derivatives of the likelihood with respect to the variance-covariance components. The Eigen decomposition of the relationship matrix proposed by Lee and Van Der Werf (2016) was included in the Newton-Raphson algorithm to improve time efficiency. Additionally, the popular pin function to estimate standard errors for linear combinations of variance components (i.e. heritabilities and genetic correlations) was added to the package as well.

Please refer to the canonical papers listed in the Literature section to check how the algorithms work. We have tested widely the methods to make sure they provide the same solution when the likelihood behaves

well but for complex problems they might lead to slightly different answers. If you have any concern please contact me at `cova_ruber@live.com.mx`.

In the following section we will go in detail over several examples on how to use mixed models in univariate and multivariate case and their use in quantitative genetics.

SECTION 2: Topics in quantitative genetics

1) Marker and non-marker based heritability calculation

The heritability is one of the most popular parameters among the breeding and genetics community because of the insight that provides in the inheritance of the trait. The heritability is usually estimated as narrow sense (h^2 ; only additive variance in the numerator σ_A^2), and broad sense (H^2 ; all genetic variance in the numerator σ_G^2).

In a classical breeding experiment with no molecular markers, special designs are performed to estimate and dissect the additive (σ_A^2) and non-additive (i.e. dominance σ_D^2) variance along with environmental variability. Designs such as generation analysis, North Carolina designs are used to dissect σ_A^2 and σ_D^2 to estimate the narrow sense heritability (h^2). When no special design is available we can still dissect the genetic variance (σ_G^2) and estimate the broad sense heritability. In this first example we will show the broad sense estimation which doesn't use covariance structures for the genotypic effect (i.e. genomic or additive relationship matrices). For big models with no covariance structures, sommer's direct inversion is a bad idea to use but we will show anyways how to do it, but keep in mind that for very sparse models we recommend using the `lmer` function from the `lme4` package or any other package using MME-based algorithms (i.e. `asreml-R`).

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(sommer)
data(DT_example)
DT <- DT_example
A <- A_example

ans1 <- mmer(Yield~1,
             random= ~ Name + Env + Env:Name + Env:Block,
             rcov= ~ units,
             data=DT)
```

```
## iteration   LogLik      wall   cpu(sec)  restrained
##      1      -40.765  14:10:51      0           0
##      2      -30.2657  14:10:51      0           0
##      3      -25.8227  14:10:51      0           1
##      4      -24.7277  14:10:51      0           1
##      5      -24.7203  14:10:51      0           1
##      6      -24.7202  14:10:51      0           1
```

```
summary(ans1)$varcomp
```

```
##              VarComp  VarCompSE   Zratio Constraint
## Name.Yield-Yield    3.718279  1.6959834  2.1924029   Positive
## Env.Yield-Yield     12.008450  12.2771178  0.9781164   Positive
## Env:Name.Yield-Yield  5.152643  1.4923912  3.4526091   Positive
## Env:Block.Yield-Yield 0.000000  0.1156675  0.0000000   Positive
## units.Yield-Yield   4.366189  0.6573086  6.6425245   Positive
```

```
(n.env <- length(levels(DT$Env)))
```

```
## [1] 3
```

```
pin(ans1, h2 ~ V1 / ( V1 + (V3/n.env) + (V5/(2*n.env)) ) )
```

```
##      Estimate      SE
## h2 0.6032715 0.1344582
```

Recently with markers becoming cheaper, thousand of markers can be run in the breeding materials. When markers are available, an special design is not necessary to dissect the additive genetic variance. The availability of the additive, dominance and epistatic relationship matrices allow us to estimate σ_A^2 , σ_D^2 and σ_I^2 , although given that A, D and E are not orthogonal the interpretation of models that fit more than A and D become 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 $\sigma_A^2 = \sigma_g^2$)

```
data(DT_cpdata)
DT <- DT_cpdata
GT <- GT_cpdata
MP <- MP_cpdata
DT$id <- DT$id; DT$ide <- DT$id
### look at the data
A <- A.mat(GT) # additive relationship matrix
D <- D.mat(GT) # dominance relationship matrix
E <- E.mat(GT) # epistatic relationship matrix
ans.ADE <- mmer(color~1,
                random=~vs(id,Gu=A) + vs(idd,Gu=D),
                rcov=~units,
                data=DT)
```

```
## iteration   LogLik      wall   cpu(sec)  restrained
##      1      -123 14:10:52      0          0
##      2     -107.864 14:10:52      0          0
##      3     -103.867 14:10:52      0          0
##      4     -103.315 14:10:52      0          0
##      5     -103.294 14:10:52      0          0
##      6     -103.293 14:10:53      1          0
```

```
(summary(ans.ADE)$varcomp)
```

```
##              VarComp  VarCompSE  Zratio Constraint
## u:id.color-color 0.003662202 0.0012194130 3.003250  Positive
## u:idd.color-color 0.001820079 0.0007406216 2.457502  Positive
## units.color-color 0.002106929 0.0002864724 7.354736  Positive
```

```
pin(ans.ADE, h2 ~ (V1) / ( V1+V3 ) )
```

```
##      Estimate      SE
## h2 0.6347926 0.08840488
```

```
pin(ans.ADE, h2 ~ (V1+V2) / ( V1+V2+V3 ) )
```

```
##      Estimate      SE
## h2 0.7223783 0.05563774
```

In the previous example we showed how to estimate the additive (σ_A^2), dominance (σ_D^2), and epistatic (σ_I^2) variance components based on markers and estimate broad (H^2) and narrow sense heritability (h^2). Notice that we used the `vs()` function which indicates that the random effect inside the parenthesis (i.e. `id`, `idd` or

ide) has a covariance matrix (A, D, or E), that will be specified in the Gu argument of the vs() function. Please DO NOT provide the inverse but 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 *GCA*₂ and *SCA* specifying the variance structure.

```
data(DT_cornhybrids)
DT <- DT_cornhybrids
DTi <- DTi_cornhybrids
GT <- GT_cornhybrids
### fit the model
modFD <- mmer(Yield~1,
              random=~ vs(at(Location,c("3","4")),GCA2),
              rcov= ~ vs(ds(Location),units),
              data=DT)
```

```
## iteration   LogLik      wall   cpu(sec)  restrained
##      1      -190.104  14:10:53      0           0
##      2      -171.543  14:10:53      0           0
##      3      -165.319  14:10:53      0           0
##      4      -164.691  14:10:53      0           0
##      5      -164.684  14:10:53      0           0
##      6      -164.684  14:10:53      0           0
```

```
summary(modFD)
```

```
## =====
##           Multivariate Linear Mixed Model fit by REML
## ***** sommer 4.1 *****
## =====
##           logLik      AIC      BIC Method Converge
## Value -164.6839 331.3677 335.3592      NR      TRUE
## =====
## Variance-Covariance components:
##           VarComp VarCompSE Zratio Constraint
## 3:GCA2.Yield-Yield      62.48      53.45  1.169  Positive
## 4:GCA2.Yield-Yield      97.99      79.56  1.232  Positive
## 1:units.Yield-Yield     216.82      30.77  7.047  Positive
## 2:units.Yield-Yield     216.82      30.77  7.047  Positive
## 3:units.Yield-Yield     493.05      77.27  6.381  Positive
## 4:units.Yield-Yield     711.98     111.63  6.378  Positive
## =====
## Fixed effects:
##   Trait      Effect Estimate Std.Error t.value
## 1 Yield (Intercept)      138.1      0.9442  146.3
## =====
## Groups and observations:
##           Yield
## 3:GCA2      20
## 4:GCA2      20
```

```
## =====
## Use the '$' sign to access results and parameters
```

In the previous example we showed how the `at()` function is used in the `mmer` solver. By using the `at` function you can specify that i.e. the GCA2 has a different variance in different Locations, in this case locations 3 and 4, but also a main GCA variance. This is considered a CS + DIAG (compound symmetry + diagonal) model.

In addition, other functions can be added on top to fit models with covariance structures, i.e. the `Gu` argument from the `vs()` function to indicate a covariance matrix (A, pedigree or genomic relationship matrix)

```
data(DT_cornhybrids)
DT <- DT_cornhybrids
DTi <- DTi_cornhybrids
GT <- GT_cornhybrids
GT[1:4,1:4]
```

```
##           A258           A634           A641           A680
## A258  2.23285528 -0.3504778 -0.04756856 -0.32239362
## A634 -0.35047780  1.4529169  0.45203869 -0.02293680
## A641 -0.04756856  0.4520387  1.96940221 -0.09896791
## A680 -0.32239362 -0.0229368 -0.09896791  1.65221984
```

```
### fit the model
modFD <- mmer(Yield-1,
              random=~ vs(at(Location,c("3","4")),GCA2,Gu=GT),
              rcov= ~ vs(ds(Location),units),
              data=DT)
```

```
## iteration   LogLik      wall    cpu(sec)   restrained
##      1      -191.286   14:10:54      0           0
##      2      -172.247   14:10:54      0           0
##      3      -165.948   14:10:54      0           0
##      4      -165.248   14:10:54      0           0
##      5      -165.23    14:10:54      0           0
##      6      -165.229   14:10:54      0           0
##      7      -165.229   14:10:54      0           0
```

```
summary(modFD)
```

```
## =====
##           Multivariate Linear Mixed Model fit by REML
## ***** sommer 4.1 *****
## =====
##           logLik      AIC      BIC Method Converge
## Value -165.2286 332.4571 336.4486      NR      TRUE
## =====
## Variance-Covariance components:
##           VarComp VarCompSE Zratio Constraint
## 3:GCA2.Yield-Yield  26.64      26.16 1.0185 Positive
## 4:GCA2.Yield-Yield  37.51      37.78 0.9927 Positive
## 1:units.Yield-Yield 216.77      30.75 7.0489 Positive
## 2:units.Yield-Yield 216.77      30.75 7.0489 Positive
## 3:units.Yield-Yield 503.62      77.87 6.4673 Positive
## 4:units.Yield-Yield 738.86     114.17 6.4715 Positive
## =====
## Fixed effects:
## Trait      Effect Estimate Std.Error t.value
```

```
## 1 Yield (Intercept)    138.1    0.9147    151
## =====
## Groups and observations:
##      Yield
## 3:GCA2    20
## 4:GCA2    20
## =====
## Use the '$' sign to access results and parameters
```

3) Using the pin calculator

Sometimes the user needs to calculate ratios or functions of specific variance-covariance components and obtain the standard error for such parameters. Examples of these are the genetic correlations, heritabilities, etc. Using the CPdata we will show how to estimate the heritability and the standard error using the pin function that uses the delta method to come up with these parameters. This can be extended for any linear combination of the variance components.

```
data(DT_cpdata)
DT <- DT_cpdata
GT <- GT_cpdata
MP <- MP_cpdata
### look at the data
A <- A.mat(GT) # additive relationship matrix
ans <- mmer(color~1,
            random=~vs(id,Gu=A),
            rcov=~units,
            data=DT)
```

3.1) Standar error for heritability

```
## iteration    LogLik      wall    cpu(sec)  restrained
##      1      -137.304  14:10:55      0           0
##      2      -115.507  14:10:55      0           0
##      3      -111.236  14:10:55      0           0
##      4      -110.755  14:10:55      0           0
##      5      -110.741  14:10:55      0           0
##      6      -110.741  14:10:55      0           0
```

```
(summary(ans.ADE)$varcomp)
```

```
##              VarComp  VarCompSE  Zratio Constraint
## u:id.color-color  0.003662202  0.0012194130  3.003250   Positive
## u:idd.color-color  0.001820079  0.0007406216  2.457502   Positive
## units.color-color  0.002106929  0.0002864724  7.354736   Positive
```

```
pin(ans, h2 ~ (V1) / ( V1+V2) )
```

```
##      Estimate      SE
## h2 0.6512157 0.06107574
```

The same can be used for multivariate models. Please check the documentation of the pin function to see more examples.

```

data(DT_btdata)
DT <- DT_btdata
mix3 <- mmer(cbind(tarsus, back) ~ sex,
             random = ~ vs(dam, Gtc=unsm(2)) + vs(fosternest,Gtc=diag(2)),
             rcov=~vs(units,Gtc=unsm(2)),
             data = DT)

```

3.2) Standar error for genetic correlation

```

## iteration      LogLik      wall      cpu(sec)  restrained
##      1      -934.205  14:11:1         6           0
##      2      -745.644  14:11:6         11          0
##      3      -661.568  14:11:9         14          0
##      4      -652.061  14:11:13        18          0
##      5      -651.67   14:11:18        23          0
##      6      -651.605  14:11:22        27          0
##      7      -651.591  14:11:27        32          0
##      8      -651.587  14:11:31        36          0
##      9      -651.586  14:11:36        41          0

```

```
summary(mix3)
```

```

## =====
##           Multivariate Linear Mixed Model fit by REML
## ***** sommer 4.1 *****
## =====
##           logLik      AIC      BIC Method Converge
## Value -651.5865 1315.173 1347.646      NR      TRUE
## =====
## Variance-Covariance components:
##           VarComp VarCompSE Zratio Constraint
## u:dam.tarsus-tarsus      0.21847  0.04743  4.606  Positive
## u:dam.tarsus-back      -0.03618  0.02644 -1.369  Unconstr
## u:dam.back-back      0.05973  0.03073  1.944  Positive
## u:fosternest.tarsus-tarsus  0.07304  0.02891  2.526  Positive
## u:fosternest.back-back  0.13158  0.03890  3.383  Positive
## u:units.tarsus-tarsus  0.56699  0.03082 18.397  Positive
## u:units.tarsus-back  -0.03004  0.02581 -1.164  Unconstr
## u:units.back-back      0.80494  0.04361 18.459  Positive
## =====
## Fixed effects:
##      Trait      Effect Estimate Std.Error t.value
## 1 tarsus (Intercept) -0.40631  0.06720 -6.0466
## 2 back (Intercept) -0.01459  0.06489 -0.2248
## 3 tarsus sexMale  0.76905  0.05711 13.4670
## 4 back sexMale  0.01057  0.06704  0.1577
## 5 tarsus sexUNK  0.21231  0.12665  1.6763
## 6 back sexUNK  0.09976  0.14794  0.6743
## =====
## Groups and observations:
##           tarsus back
## u:dam      106  106
## u:fosternest 104  104
## =====

```



```
## Use the '$' sign to access results and parameters
```

```
##### calculate the genetic correlation  
pin(mix3, gen.cor ~ V2 / sqrt(V1*V3))
```

```
##           Estimate      SE  
## gen.cor -0.3167271 0.2228247
```

4) 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 if reciprocal and self cross (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 phenotypic 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 <- mmer(Yield~Location,  
              random=~GCA1+GCA2+SCA,  
              rcov=~units,  
              data=DT)
```

```
## iteration   LogLik      wall   cpu(sec)  restrained  
##      1      -149.436  14:11:37      0           0  
##      2      -136.475  14:11:37      0           1  
##      3      -132.852  14:11:37      0           1  
##      4      -132.625  14:11:37      0           1  
##      5      -132.596  14:11:37      0           1  
##      6      -132.59  14:11:37      0           1  
##      7      -132.589  14:11:37      0           1  
##      8      -132.589  14:11:37      0           1
```

```
(suma <- summary(modFD)$varcomp)
```

```
##           VarComp VarCompSE      Zratio Constraint  
## GCA1.Yield-Yield  0.000000  16.50337  0.0000000 Positive  
## GCA2.Yield-Yield  7.412226  18.94200  0.3913116 Positive  
## SCA.Yield-Yield  187.560303  41.59428  4.5092817 Positive  
## units.Yield-Yield 221.142463  18.14716 12.1860656 Positive
```

```
Vgca <- sum(suma[1:2,1])  
Vsca <- suma[3,1]  
Ve <- suma[4,1]  
Va = 4*Vgca
```

```
Vd = 4*Vsca
Vg <- Va + Vd
(H2 <- Vg / (Vg + (Ve)) )
```

```
## [1] 0.7790856
```

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

```
## [1] 0.02961832
```

Don't worry too much about the small 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 this second data set we show a small half diallel with 7 parents crossed in one direction. $n(n-1)/2$ crosses are possible $7(6)/2 = 21$ unique crosses. Parents appear as males or females indistinctly. 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 ($\sigma_G^2 CA$ and $\sigma_S^2 CA$ respectively), and BLUPs for GCA and SCA of the parents can be extracted. We would show first how to use it with the `mmer` function using the `overlay()` function. The specific model here is:

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

```
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
```

```
DT$femalef <- as.factor(DT$female)
DT$malef <- as.factor(DT$male)
DT$genof <- as.factor(DT$geno)
#### model using overlay
modh <- mmer(sugar~1,
             random=-vs(overlay(femalef,malef))
             + genof,
             data=DT)
```

```
## iteration   LogLik      wall   cpu(sec)  restrained
##      1      -10.425  14:11:38      0          0
##      2      -6.487  14:11:38      0          0
##      3      -5.732  14:11:38      0          0
##      4      -5.67494  14:11:38      0          0
##      5      -5.67441  14:11:38      0          0
```

```
summary(modh)$varcomp
```

```
##              VarComp VarCompSE  Zratio Constraint
## u:femalef.sugar-sugar 5.507899 3.5741151 1.541052  Positive
## genof.sugar-sugar    1.815784 1.3629575 1.332238  Positive
## units.sugar-sugar    3.117538 0.9626094 3.238632  Positive
```

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.

5) Genomic selection: predicting mendelian sampling

In this section we will use wheat data from CIMMYT to show how is genomic selection 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.

```
data(DT_wheat)
DT <- DT_wheat
GT <- GT_wheat
colnames(DT) <- paste0("X",1:ncol(DT))
DT <- as.data.frame(DT);DT$id <- 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)
```

```
##           X1           X2           X3           X4  id
## 775  1.6716295 -1.72746986 -1.89028479  0.0509159  775
## 2166 -0.2527028  0.40952243  0.30938553 -1.7387588 2166
## 2167           NA -0.64862633 -0.79955921 -1.0535691 2167
## 2465  0.7854395  0.09394919  0.57046773  0.5517574 2465
## 3881  0.9983176 -0.28248062  1.61868192 -0.1142848 3881
## 3889  2.3360969  0.62647587  0.07353311  0.7195856 3889
```

```
## GBLUP
ans <- mmer(X1~1,
           random=~vs(id,Gu=K),
           rcov=~units,
           data=y.trn) # kinship based
```

```
## iteration  LogLik      wall    cpu(sec)  restrained
##      1      -207.28  14:11:39         0           0
##      2      -204.823  14:11:39         0           0
##      3      -203.988  14:11:39         0           0
##      4      -203.866  14:11:39         0           0
##      5      -203.862  14:11:39         0           0
##      6      -203.861  14:11:39         0           0
```

```
ans$U$`u:id`$X1 <- as.data.frame(ans$U$`u:id`$X1)
rownames(ans$U$`u:id`$X1) <- gsub("id","",rownames(ans$U$`u:id`$X1))
cor(ans$U$`u:id`$X1[vv,],DT[vv,"X1"], use="complete")
```

```
## [1] 0.5737594
```

```
## rrBLUP
ans2 <- mmer(X1~1,
```

```

random=~vs(list(GT)),
rcov=~units,
data=y.trn) # kinship based

```

```

## iteration    LogLik      wall    cpu(sec)  restrained
##      1      -351.655  14:11:41      1          0
##      2      -249.864  14:11:41      1          0
##      3      -214.384  14:11:41      1          0
##      4      -204.423  14:11:41      1          0
##      5      -203.883  14:11:41      1          0
##      6      -203.862  14:11:41      1          0
##      7      -203.861  14:11:42      2          0

```

```

u <- GT %*% as.matrix(ans2$U$`u:GT`$X1) # BLUPs for individuals
rownames(u) <- rownames(GT)
cor(u[vv,],DT[vv,"X1"]) # same correlation

```

```
## [1] 0.5737681
```

```
# the same can be applied in multi-response models in GBLUP or rrBLUP
```

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:

$$u_1 \sim N(0, K_1\sigma_u^2 1)$$

$$u_2 \sim N(0, K_2\sigma_u^2 2)$$

$$u_s \sim N(0, K_3\sigma_u^2 s)$$

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 occurred 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
Mf <- Mf_technow
Ad <- Ad_technow
Af <- Af_technow
# RUN THE PREDICTION MODEL
y.trn <- DT
vv1 <- which(!is.na(DT$GY))
vv2 <- sample(vv1, 100)
y.trn[vv2,"GY"] <- NA
anss2 <- mmer(GY~1,
              random=~vs(dent,Gu=Ad) + vs(flint,Gu=Af),
              rcov=~units,
              data=y.trn)

```

```
## iteration      LogLik      wall      cpu(sec)    restrained
##      1         104.311    14:11:44         2           0
##      2         143.766    14:11:45         3           0
##      3         152.662    14:11:46         4           0
##      4         153.91     14:11:47         5           0
##      5         153.974    14:11:48         6           0
##      6         153.978    14:11:49         7           0
##      7         153.978    14:11:50         8           0
```

```
summary(anss2)$varcomp
```

```
##              VarComp VarCompSE      Zratio Constraint
## u:dent.GY-GY 16.06423 2.5737578 6.241548  Positive
## u:flint.GY-GY 11.42070 2.1591718 5.289390  Positive
## units.GY-GY 16.81801 0.7689509 21.871368  Positive
```

```
zu1 <- model.matrix(~dent-1,y.trn) %*% anss2$U$`u:dent`$GY
zu2 <- model.matrix(~flint-1,y.trn) %*% anss2$U$`u:flint`$GY
u <- zu1+zu2+anss2$Beta[1,"Estimate"]
cor(u[vv2,], DT$GY[vv2])
```

```
## [1] 0.7756383
```

In the previous model we only used the GCA effects (GCA1 and GCA2) for practicality, although it's been shown that the SCA effect doesn't actually help that much in increasing prediction accuracy and 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 10578x10578 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.

7) 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 design a necessity more than anything else. Experimental designs such as augmented designs and partially-replicated (p-rep) designs become every day more common this 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
GT <- GT_cpdata
MP <- MP_cpdata
### mimic two fields
A <- A.mat(GT)
mix <- mmer(Yield~1,
            random=~vs(id, Gu=A) +
              vs(Rowf) +
```

```

vs(Colf) +
vs(spl2D(Row,Col)),
rcov=~vs(units),
data=DT)

```

```

## iteration    LogLik      wall    cpu(sec)  restrained
##      1      -154.198  14:11:51      0          0
##      2      -152.064  14:11:51      0          0
##      3      -151.265  14:11:52      1          0
##      4      -151.202  14:11:52      1          0
##      5      -151.201  14:11:52      1          0

```

```
summary(mix)
```

```

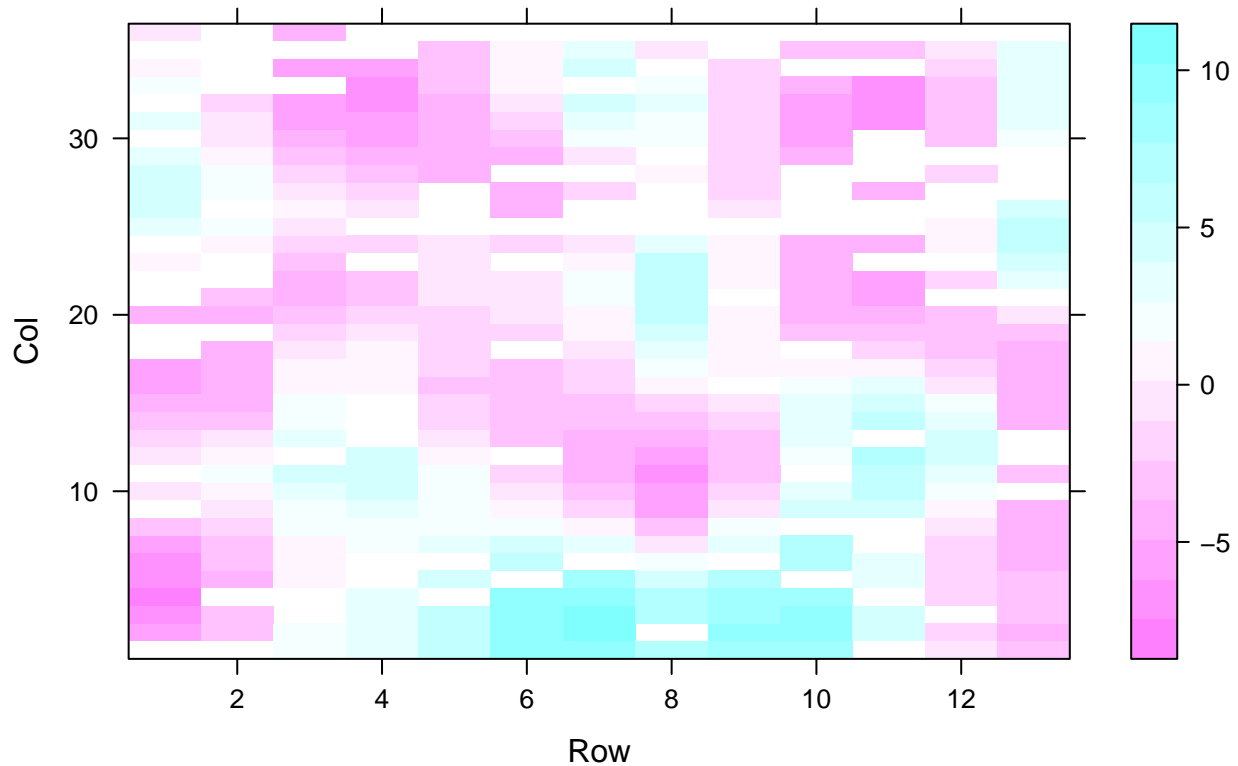
## =====
##           Multivariate Linear Mixed Model fit by REML
## ***** sommer 4.1 *****
## =====
##           logLik      AIC      BIC Method Converge
## Value -151.2011 304.4021 308.2938      NR      TRUE
## =====
## Variance-Covariance components:
##           VarComp VarCompSE Zratio Constraint
## u:id.Yield-Yield      783.4      319.3 2.4536 Positive
## u:Rowf.Yield-Yield      814.7      390.5 2.0863 Positive
## u:Colf.Yield-Yield      182.2      129.7 1.4053 Positive
## u:Row.Yield-Yield      513.6      694.7 0.7393 Positive
## u:units.Yield-Yield  2922.6      294.1 9.9368 Positive
## =====
## Fixed effects:
## Trait      Effect Estimate Std.Error t.value
## 1 Yield (Intercept)      132.1      8.791  15.03
## =====
## Groups and observations:
##           Yield
## u:id      363
## u:Rowf    13
## u:Colf    36
## u:Row     168
## =====
## Use the '$' sign to access results and parameters

```

```

# make a plot to observe the spatial effects found by the spl2D()
W <- with(DT,spl2D(Row,Col)) # 2D spline incidence matrix
DT$spatial <- W%*%mix$U$`u:Row`$Yield # 2D spline BLUPs
lattice::levelplot(spatial~Row*Col, data=DT) # plot the spatial effect by row and column

```



Notice that the job is done by the `sp12D()` function that takes the Row and Col information to fit a spatial kernel.

8) Multivariate genetic models and genetic correlations

Sometimes is important to estimate genetic variance-covariance among traits, multi-reponse models are very useful for such task. Let see an example with 3 traits (color, Yield, and Firmness) 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
A <- A.mat(GT)
ans.m <- mmer(cbind(Yield,color)~1,
              random=~ vs(id, Gu=A, Gtc=unsm(2))
              + vs(Rowf,Gtc=diag(2))
              + vs(Colf,Gtc=diag(2)),
              rcov=~ vs(units, Gtc=unsm(2)),
              data=DT)
```

##	iteration	LogLik	wall	cpu(sec)	restrained
##	1	-375.872	14:11:54	1	0
##	2	-291.932	14:11:54	1	0
##	3	-258.273	14:11:55	2	0
##	4	-253.459	14:11:56	3	0
##	5	-253.291	14:11:56	3	0
##	6	-253.278	14:11:57	4	0
##	7	-253.277	14:11:57	4	0

```
##      8      -253.277   14:11:58      5      0
```

Now you can extract the BLUPs using the 'randef' function or simple accessing with the '\$' sign and pick 'u.hat'. Also, genetic correlations and heritabilities can be calculated easily.

```
cov2cor(ans.m$sigma$`u:id`)
```

```
##          Yield      color
## Yield 1.0000000 0.1234441
## color 0.1234441 1.0000000
```

SECTION 3: Special topics in Quantitative genetics

1) Partitioned model

The partitioned model was popularized by () to show that marker effects can be obtained fitting a GBLUP model to reduce the computational burden and then recover them by creating some special matrices MM' for GBLUP and $M'(M'M)^{-1}$ to recover marker effects. Here we show a very easy example using the DT_cpdata:

```
library(sommer)
data("DT_cpdata")
DT <- DT_cpdata
M <- GT_cpdata

#####
# MARKER MODEL
#####
mix.marker <- mmer(color~1,
                  random=~Rowf+vs(M),
                  rcov=~units,data=DT,
                  verbose = FALSE)

me.marker <- mix.marker$U$`u:M`$color

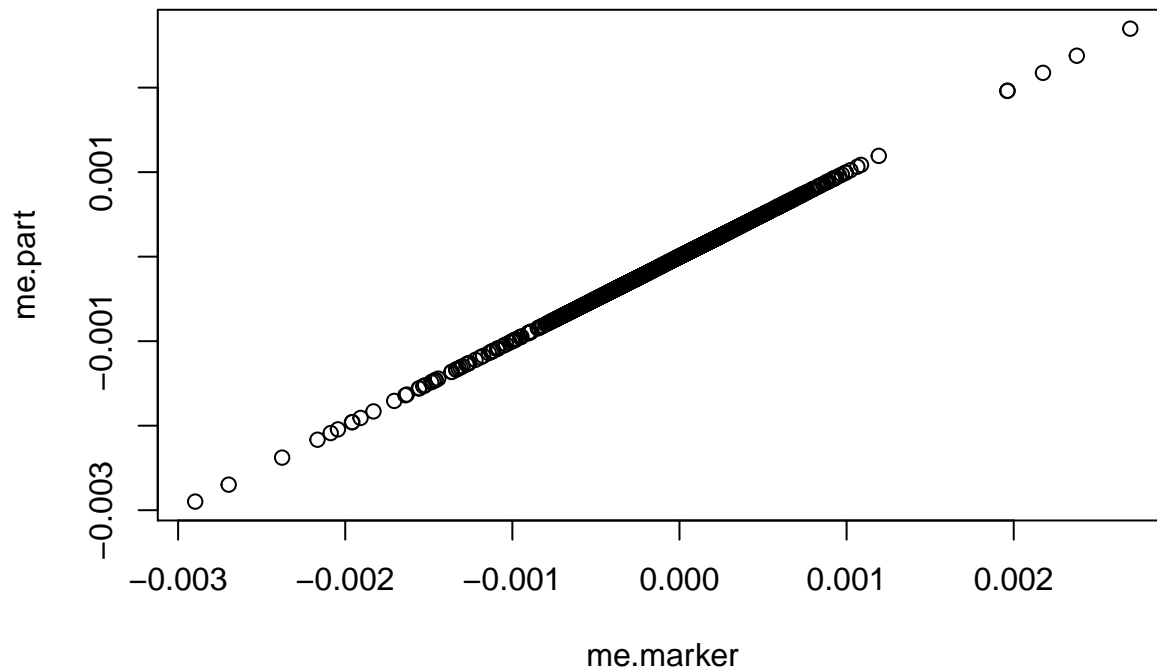
#####
# PARTITIONED GBLUP MODEL
#####

MMT<-M%*%t(M) ## additive relationship matrix
MMTinv<-solve(MMT) ## inverse
MTMMTinv<-t(M)%*%MMTinv # M' %*% (M'M)-

mix.part <- mmer(color~1,
                random=~Rowf+vs(id, Gu=MMT),
                rcov=~units,data=DT,
                verbose = FALSE)

#convert BLUPs to marker effects me=M'(M'M)- u
me.part<-MTMMTinv%*%matrix(mix.part$U$`u:id`$color,ncol=1)

# compare marker effects between both models
plot(me.marker,me.part)
```

As can be seen this 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$eigenvectors
D<-diag(UD$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)

modeld <- mmer(cbind(X1,X2) ~ UX - 1,
              random = ~vs(id,Gu=D),
              rcov = ~vs(units),
              data=DTd, verbose = FALSE)
```

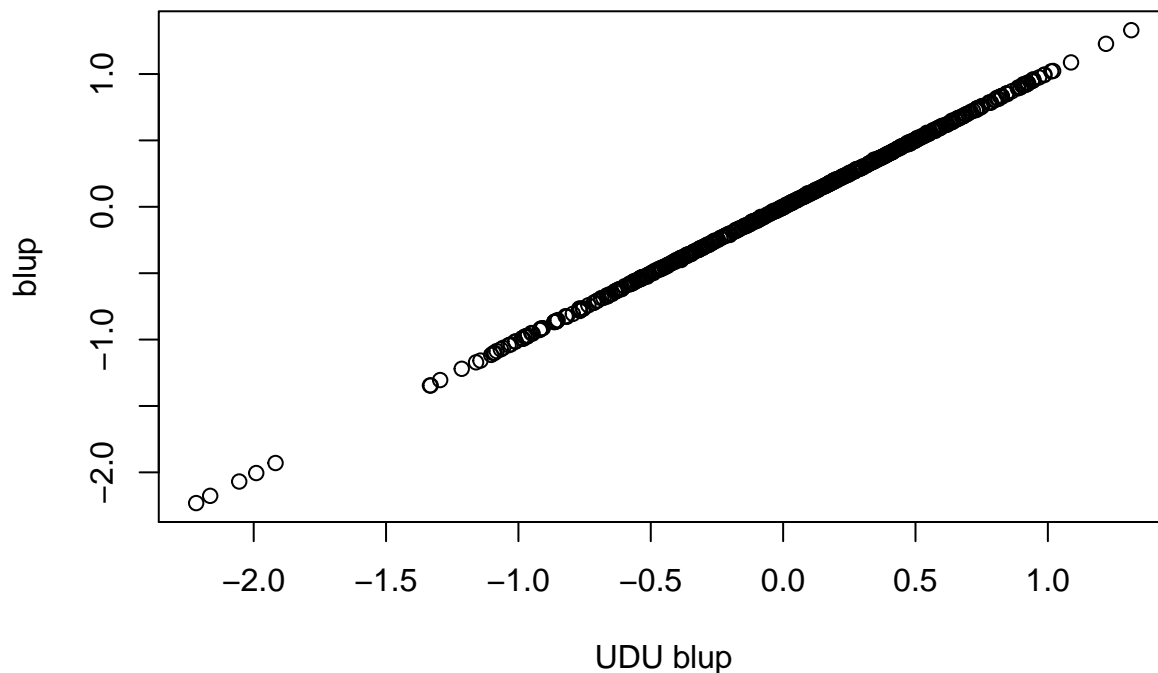
```

# dataset for normal model
DTn<-data.frame(id = rownames(G) , DT_wheat)
DTn$id<-as.character(DTn$id)

modeln <- mmer(cbind(X1,X2) ~ 1,
              random = ~vs(id,Gu=G),
              rcov = ~vs(units),
              data=DTn, verbose = FALSE)

## compare regular and transformed blups
plot(x=(solve(t(U)))%*%modeln$U$`u:id`$X2[colnames(D)],
     y=modeln$U$`u:id`$X2[colnames(D)], xlab="UDU blup",
     ylab="blup")

```



As can be seen the two models are equivalent. Despite that sommer 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

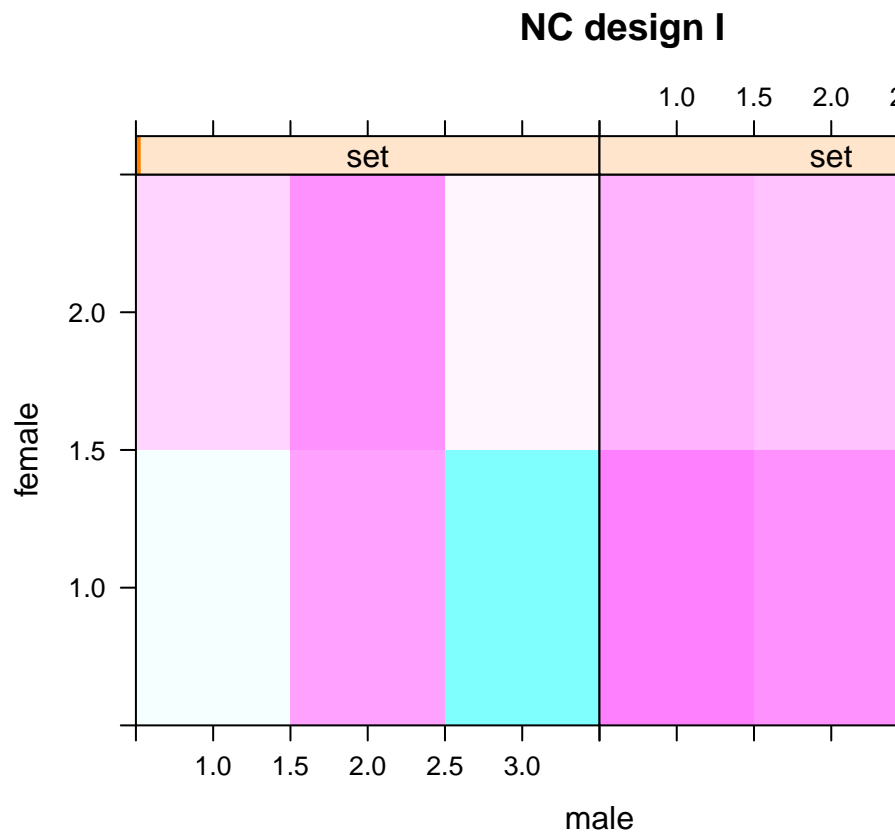
Estimating variance components has been a topic of interest for the breeding community for a long time. Here we show how to calculate additive and dominance variance using the North Carolina Design I (Nested design) and North Carolina Design II (Factorial design) using the classical Expected Mean Squares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

```

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")
```



North Carolina Design I (Nested design)

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

```
## 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
```

```
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))
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

#####
## REML method
#####
mix2 <- mmer(yield~ setf + setf:repf,
             random=~femalef:malef:setf + malef:setf,
             data=DT, verbose = FALSE)
vc <- summary(mix2)$varcomp; vc

##                               VarComp  VarCompSE   Zratio Constraint
## femalef:malef:setf.yield-yield 0.08056338 0.08096526 0.9950364   Positive
## malef:setf.yield-yield          0.39480593 0.32832346 1.2024908   Positive
## units.yield-yield               0.10691762 0.04785610 2.2341480   Positive

Vfm <- vc[1,"VarComp"]
Vm <- vc[2,"VarComp"]

## 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
## 1.579224 -1.256970

```

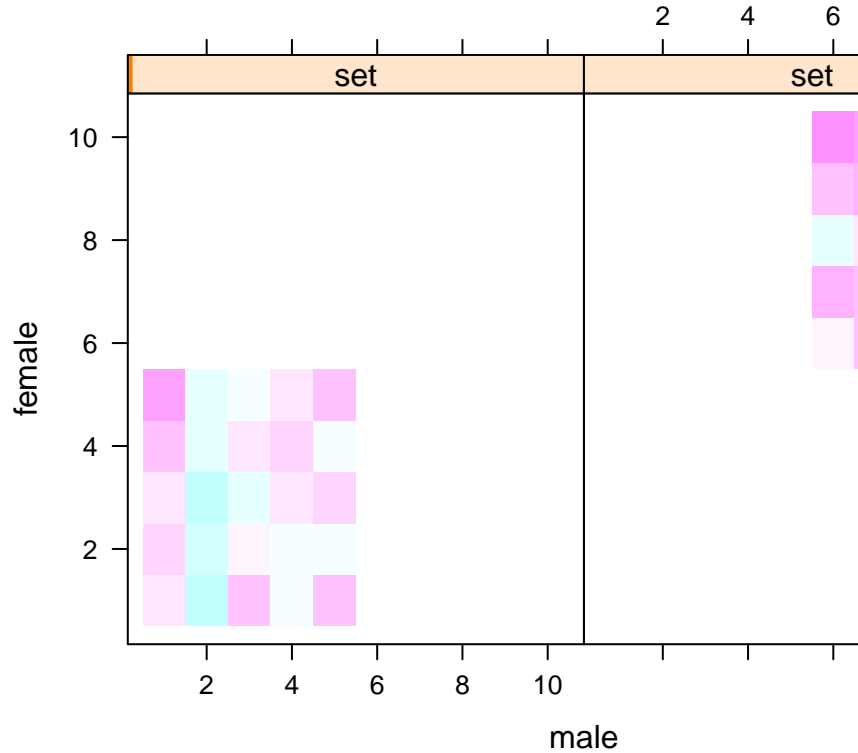
As can be seen the REML method is easier than manipulating the MS and we arrive to the same results.

```

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")

```

NC desing II



North Carolina Design II (Factorial design)

`head(DT)`

```
##   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
```

```
N=with(DT,table(female, male, set))
```

```
nmale=length(which(N[, ,1] > 0))
```

```
nfemale=length(which(N[, 1,1] > 0))
```

```
nrep=table(N[, ,1])
```

```
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
```

```
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))
nmale=length(unique(DT$male))
Vfm <- (ms3-mse)/nrep;
Vf <- (ms2-ms3)/(nrep*nmale);
Vm <- (ms1-ms3)/(nrep*nfem);

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
```

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

```
mix2 <- mmer(yield~ setf + setf:repf ,
             random=-femalef:malef:setf + malef:setf + femalef:setf,
             data=DT)
```

```
## iteration   LogLik      wall    cpu(sec)   restrained
##      1      -47.2423   14:12:26      0           0
##      2      -46.9375   14:12:26      0           0
##      3      -46.8452   14:12:26      0           0
##      4      -46.8361   14:12:26      0           0
##      5      -46.836   14:12:26      0           0
```

```
vc <- summary(mix2)$varcomp; vc
```

##	VarComp	VarCompSE	Zratio	Constraint
## femalef:malef:setf.yield-yield	2215.618	2284.794	0.9697231	Positive
## malef:setf.yield-yield	5493.338	3610.989	1.5212836	Positive
## femalef:setf.yield-yield	2710.176	2236.621	1.2117280	Positive
## units.yield-yield	18580.739	2681.742	6.9286068	Positive

```
Vfm <- vc[1, "VarComp"]
Vm <- vc[2, "VarComp"]
Vf <- vc[3, "VarComp"]

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
## 10840.704  8862.471
```

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

4) Dominance variance

The estimation of non-additive variance has been proposed to be a challenge since the additive and dominance relationship matrices are not orthogonal. In recent literature it has been proposed that the best practice to fit the dominance component is to fit the additive component first and then fix the value of that variance c

```
data(DT_cpdata)
DT <- DT_cpdata
GT <- GT_cpdata
MP <- MP_cpdata
#### create the variance-covariance matrix
A <- A.mat(GT) # additive relationship matrix
#### look at the data and fit the model
mix1 <- mmer(Yield~1,
             random=~vs(id,Gu=A),
             rcov=~units,
             data=DT, verbose = FALSE)

####=====####
#### adding dominance and forcing the other VC's
####=====####
DT$idd <- DT$id;
D <- D.mat(GT) # dominance relationship matrix
mm <- matrix(3,1,1) ## matrix to fix the var comp

mix2 <- mmer(Yield~1,
             random=~vs(id, Gu=A, Gti=mix1$sigma_scaled`u:id`, Gtc=mm)
             + vs(idd, Gu=D, Gtc=unsm(1)),
             rcov=~vs(units,Gti=mix1$sigma_scaled$units, Gtc=mm),
             data=DT, verbose = FALSE)

# analyze variance components
summary(mix1)$varcomp
```

```
##          VarComp VarCompSE    Zratio Constraint
## u:id.Yield-Yield  650.4145  325.5562  1.997856   Positive
## units.Yield-Yield 4031.0153  344.6051 11.697493   Positive
```

```
summary(mix2)$varcomp
```

```
##          VarComp VarCompSE    Zratio Constraint
## u:id.Yield-Yield  650.4145  504.0361  1.2904126    Fixed
## u:idd.Yield-Yield  220.6311  410.7679  0.5371186    Positive
## u:units.Yield-Yield 4031.0153  360.7322 11.1745357    Fixed
```

Final remarks

Keep in mind that sommer uses direct inversion (DI) algorithm which can be very slow for large datasets. The package is focused in problems of the type $p > n$ (more random effect levels than observations) and models with dense covariance structures. For example, for experiment with dense covariance structures with low-replication (i.e. 2000 records from 1000 individuals replicated twice with a covariance structure of 1000x1000) sommer will be faster than MME-based software. Also for genomic problems with large number of random effect levels, i.e. 300 individuals (n) with 100,000 genetic markers (p). For highly replicated trials with small covariance structures or $n > p$ (i.e. 2000 records from 200 individuals replicated 10 times with covariance structure of 200x200) asreml or other MME-based algorithms will be much faster and we recommend you to opt for those software. When datasets are big, the installation of the OpenBLAS library can make sommer quite fast and sometimes faster than asreml given the capability of sommer to take advantage of the multi-processor architecture of some systems.

Literature

Covarrubias-Pazaran G. 2016. Genome assisted prediction of quantitative traits using the R package sommer. PLoS ONE 11(6):1-15.

Covarrubias-Pazaran G. 2018. Software update: Moving the R package sommer to multivariate mixed models for genome-assisted prediction. doi: <https://doi.org/10.1101/354639>

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.