# Quantitative genetics using the sommer package

Giovanny Covarrubias-Pazaran

2024-07-30

The sommer package was developed to provide R users with a powerful and reliable multivariate mixed model solver for different genetic and non-genetic analyses in diploid and polyploid organisms. This package allows the user to estimate variance components for a mixed model with the advantages of specifying the variance-covariance structure of the random effects, specifying heterogeneous variances, and obtaining 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 derect-inversion algorithms. Although the vignette shows examples using the mmer function, the mmec function can be faster when working with more records than coefficients to be estimated and we highly recommend to shift to the use of the `mmec` function.

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.

The 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 `vpredict()` calculator
4) Half and full diallel designs (using the overlay)
5) Genomic selection (predicting mendelian sampling)
    - GBLUP
    - rrBLUP
6) Indirect genetic effects
7) Single cross prediction (hybrid prediction)
8) Spatial modeling (using the 2-dimensional splines)
9) Multivariate genetic models and genetic correlations

**SECTION 3: Special topics in quantitative genetics**

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

## SECTION 1: Introduction

### Backgrounds in linear algebra

The core of the package is the `mmer()` and `mmec` functions which solve the mixed model equations. The functions are an interface to call the `NR` Direct-Inversion Newton-Raphson or Average Information or mme-based Average Information (Tunnicliffe 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:

$y_1 = X_1\beta_1 + Z_1u_1 + \epsilon_1$

$y_2 = X_2\beta_2 + Z_2u_2 + \epsilon_2$

...

$y_i = X_i\beta_i + Z_iu_i + \epsilon_i$

where $y_i$ is a vector of trait phenotypes, $\beta_i$ 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, ..., t). The random effects ($u_1$ ... $u_i$ 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 distributions of the multivariate response and the phenotypic variance covariance (V) are:

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

Y ~ MVN($X\beta$, V)

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

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

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

where K is the relationship or covariance matrix for the kth random effect (u=1,...,k), and R=I is an identity matrix for the residual term. The terms $\sigma^2_{g_i}$ and $\sigma^2_{\epsilon_i}$ denote the genetic (or any of the kth random terms) and residual variance of trait `i`, respectively and $\sigma_{g_{ij}}$ and $\sigma_{\epsilon_{ij}}$ the genetic (or any of the kth random terms) and residual covariance between traits `i` and `j` (i=1,...,t, and j=1,...,t). The algorithm implemented optimizes the log likelihood:

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

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

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

Where $\theta$ 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 kth cycle, $\frac{dL}{d\sigma^2_i}$ 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 `vpredict()` 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

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 $\sigma^2_A$), and broad sense ($H^2$; all genetic variance in the numerator $\sigma^2_G$).

In a classical breeding experiment with no molecular markers, special designs are performed to estimate and dissect the additive ($\sigma^2_A$) and non-additive (e.g., dominance $\sigma^2_D$, and epistatic $\sigma^2_E$) variance along with environmental variability. Designs such as generation analysis, North Carolina designs are used to dissect $\sigma^2_A$ and $\sigma^2_D$ to estimate the narrow sense heritability ($h^2$) using only $\sigma^2_A$ in the numerator. When no special design is available we can still disect the genetic variance ($\sigma^2_G$) 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). For big models with no relationship matrices, sommer's direct inversion is a bad idea to use but we will still show how to do it, but keep in mind that for very sparse models with no relationship matrices or other special covariance structures we recommend using the `lmer()` function from the lme4 package or any other package using MME-based algorithms (e.g., 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 <- mmec(Yield~1,
             random= ~ Name + Env + Env:Name + Env:Block,
             rcov= ~ units, nIters=3,
             data=DT, verbose = FALSE)
summary(ans1)$varcomp
```

```
##                         VarComp    VarCompSE       Zratio Constraint
## Name:isc:isc       5.922491e+00 3.483681e-16 1.700067e+16   Positive
## Env:isc:isc        1.115162e+02 1.191111e-14 9.362369e+15   Positive
## Env:Name:isc:isc   1.973759e+01 1.550374e-16 1.273085e+17   Positive
## Env:Block:isc:isc  1.000000e-10 1.692545e-04 5.908264e-07      Fixed
## units:isc:isc      1.000000e-10 1.098857e-05 9.100366e-06   Positive
```

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

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

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

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 $\sigma^2_A$, $\sigma^2_D$ and $\sigma^2_I$, 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
DT$idd <-DT$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=~vsr(id,Gu=A) + vsr(idd,Gu=D),
                rcov=~units, nIters=3,
                data=DT,verbose = FALSE)
(summary(ans.ADE)$varcomp)
```

```
##                       VarComp    VarCompSE   Zratio Constraint
## u:id.color-color  0.003486639 0.0011076709 3.147721   Positive
## u:idd.color-color 0.001286778 0.0005071774 2.537136   Positive
## units.color-color 0.002152054 0.0002976821 7.229372   Positive
```

```
vpredict(ans.ADE, h2 ~ (V1) / ( V1+V3) ) # narrow sense
```

```
##     Estimate         SE
## h2 0.6183417 0.08605268
```

```
vpredict(ans.ADE, h2 ~ (V1+V2) / ( V1+V2+V3) ) # broad-sense
```

```
##     Estimate         SE
## h2 0.6892552 0.05896317
```

In this example we showed how to estimate the additive ($\sigma^2_A$) and dominance ($\sigma^2_D$) variance components based on markers and estimate broad ($H^2$) and narrow-sense heritability ($h^2$). Notice that we used the `vsr()` 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 `vsr()` function. Please DO NOT provide the inverse, 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 $GCA_2$ and $SCA$ specifying the variance structure.

```
# data(DT_cornhybrids)
# DT <- DT_cornhybrids
# DTi <- DTi_cornhybrids
# GT <- GT_cornhybrids
# ### fit the model
# modFD <- mmec(Yield~1,
#               random=~ vsc(atc(Location,c("3","4")),isc(GCA2)),
#               rcov= ~ vsc(dsc(Location),isc(units)), nIters=3,
#               returnParam = F,
#               data=DT, verbose = FALSE)
# summary(modFD)
```

In the previous example we showed how the `atr()` function is used in the `mmer()` solver. By using the `atr()` 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 `vsr()` function to indicate a covariance matrix (A, pedigree or genomic relationship matrix)

```
# data(DT_cornhybrids)
# DT <- DT_cornhybrids
# DTi <- DTi_cornhybrids
# GT <- as(GT_cornhybrids, Class = "dgCMatrix")
# GT[1:4,1:4]
# DT=DT[with(DT, order(Location)), ]
# ### fit the model
# modFD <- mmec(Yield~1,
#               random=~ vsc(atc(Location,c("3","4")),isc(GCA2),Gu=GT),
#               rcov= ~ vsc(dsc(Location),isc(units)), nIters=3,
#               data=DT, verbose = FALSE)
# summary(modFD)
```

### 3) Using the vpredict calculator

Sometimes the user needs to calculate ratios or functions of specific variance-covariance components and obtain the standard errors 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 `vpredict()` 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=~vsr(id,Gu=A),
            rcov=~units, nIters=3,
```

```
                data=DT, verbose = FALSE)
(summary(ans.ADE)$varcomp)
```

### 3.1) Standar error for heritability

```
##                         VarComp    VarCompSE   Zratio Constraint
## u:id.color-color  0.003486639 0.0011076709 3.147721   Positive
## u:idd.color-color 0.001286778 0.0005071774 2.537136   Positive
## units.color-color 0.002152054 0.0002976821 7.229372   Positive
```

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

```
##     Estimate         SE
## h2 0.6405467 0.05805209
```

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

```
## just silenced to avoid too much time building the vignettes
# data(DT_btdata)
# DT <- DT_btdata
# mix3 <- mmer(cbind(tarsus, back) ~ sex,
#                random = ~ vsr(dam, Gtc=unsm(2)) + vsr(fosternest,Gtc=diag(2)),
#                rcov=~vsr(units,Gtc=unsm(2)), nIters=3,
#                data = DT, verbose = FALSE)
# summary(mix3)
# #### calculate the genetic correlation
# vpredict(mix3, gen.cor ~ V2 / sqrt(V1*V3))
```

### 3.2) Standar error for genetic correlation

### 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 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 <- mmec(Yield~Location,
```

```
              random=~GCA1+GCA2+SCA,
              rcov=~units, nIters=3,
              data=DT, verbose = FALSE)
```

```
## Updated VC is not positive definite, changing to EM step
## Update using constraints
## Updated VC is not positive definite, changing to EM step
## Update using constraints
```

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

```
##                     VarComp VarCompSE       Zratio Constraint
## GCA1:isc:isc  1.000000e-10  36.89059 2.710718e-12      Fixed
## GCA2:isc:isc  4.629216e+01  37.14297 1.246324e+00   Positive
## SCA:isc:isc   8.937350e+01  21.56529 4.144322e+00   Positive
## units:isc:isc 2.388020e+02  28.83389 8.281992e+00   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] 0.6944174
```

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

```
## [1] 0.2369507
```

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 first how to do so 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 <- mmec(sugar~1,
             random=~vsc(isc(overlay(femalef,malef)) )
             + genof, nIters=3,
             data=DT, verbose = FALSE)
summary(modh)$varcomp
```

```
##                             VarComp    VarCompSE       Zratio Constraint
## femalef:malef:isc:isc 5.780585e+00 8.287295e-17 6.975238e+16   Positive
## genof:isc:isc         2.133867e+01 7.515991e-17 2.839103e+17   Positive
## units:isc:isc         1.000000e-10 1.697717e-05 5.890264e-06   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 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.

```
data(DT_wheat)
DT <- DT_wheat
GT <- GT_wheat[,1:200]
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=~vsr(id,Gu=K),
            rcov=~units,nIters=3,
            data=y.trn, verbose = FALSE) # kinship based
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] 0.4310372
```

```
## rrBLUP
ans2 <- mmer(X1~1,
             random=~vsr(list(GT), buildGu = FALSE),
             rcov=~units, getPEV = FALSE, nIters=3,
             data=y.trn, verbose = FALSE) # kinship based

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] 0.4370181
```

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

Please notice that when specifying the marker matrix as a random effect we used the argument 'buildGu=FALSE' to inform the 'mmer' function that a covariance matrix for the levels of the random effect shouldn't be built. Imagine a model with 100,000 markers, that would imply a relationship matrix of 100,000 x 100,000. If that matrix is a diagonal it would only compromise the speed and memory of the function. By setting 'buildGu=FALSE' the mmer solver will avoid the matrix multiplications using that huge diagonal matrix. If you want to specify a relationship matrix for the marker matrix then you cannot use that 'buildGu' argument.

### 6) 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 first fit a direct genetic effects model:

```
# data(DT_ige)
# DT <- DT_ige
# Af <- A_ige
# An <- A_ige

## Direct genetic effects model
# modDGE <- mmec(trait ~ block,
#                random = ~ focal,
#                rcov = ~ units, nIters=3,
#                data = DT, verbose=FALSE)
# summary(modDGE)$varcomp
```

We now fit the indirect genetic effects model without covariance between DGE and IGE:

```
# data(DT_ige)
# DT <- DT_ige
# A <- A_ige
#
# ## Indirect genetic effects model
# modIGE <- mmec(trait ~ block, dateWarning = FALSE,
#                random = ~ focal + neighbour, verbose = FALSE,
#                rcov = ~ units, nIters=100,
```

```
#                data = DT)
# summary(modIGE)$varcomp
```

We now fit the indirect genetic effects model with covariance between DGE and IGE for which we will use the `gvsr()` function:

```
# ### Indirect genetic effects model
# modIGE <- mmec(trait ~ block, dateWarning = FALSE,
#                 random = ~ covc( vsc(isc(focal)), vsc(isc(neighbour)) ),
#                 rcov = ~ units, nIters=100, verbose = FALSE,
#                data = DT)
# summary(modIGE)$varcomp
```

On top of that we can include a relationship matrix for the two random effects that are being forced to co-vary

```
### Indirect genetic effects model
# Ai <- as( solve(A_ige + diag(1e-5, nrow(A_ige),nrow(A_ige) )), Class="dgCMatrix")
# # Indirect genetic effects model with covariance between DGE and IGE using relationship matrices
# modIGE <- mmec(trait ~ block, dateWarning = FALSE,
#                 random = ~ covc( vsc(isc(focal), Gu=Ai), vsc(isc(neighbour), Gu=Ai) ),
#                 rcov = ~ units, nIters=100, verbose = FALSE,
#                data = DT)
# summary(modIGE)$varcomp
```

### 7) 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:

$u_1 \sim \mathrm{N}(0,\ K_1\sigma^2_u 1)$

$u_2 \sim \mathrm{N}(0,\ K_2\sigma^2_u 2)$

$u_s \sim \mathrm{N}(0,\ K_3\sigma^2_u s)$

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)
Adi <- as(solve(Ad + diag(1e-4,ncol(Ad),ncol(Ad))), Class="dgCMatrix")
Afi <- as(solve(Af + diag(1e-4,ncol(Af),ncol(Af))), Class="dgCMatrix")
# RUN THE PREDICTION MODEL
y.trn <- DT
vv1 <- which(!is.na(DT$GY))
```

```
vv2 <- sample(vv1, 100)
y.trn[vv2,"GY"] <- NA
anss2 <- mmec(GY~1,
              random=~vsc(isc(dent),Gu=Adi) + vsc(isc(flint),Gu=Afi),
              rcov=~units, nIters=15,
              data=y.trn, verbose = FALSE)
summary(anss2)$varcomp
```

```
##                     VarComp VarCompSE   Zratio Constraint
## dent:Adi:isc:isc  15.86817  2.992973 5.301809   Positive
## flint:Afi:isc:isc 11.29153  3.280194 3.442337   Positive
## units:isc:isc     15.97828  2.831564 5.642916   Positive
```

```
# zu1 <- model.matrix(~dent-1,y.trn) %*% anss2$uList$`vsc(isc(dent), Gu = Adi)`
# zu2 <- model.matrix(~flint-1,y.trn) %*% anss2$uList$`vsc(isc(flint), Gu = Afi)`
# u <- zu1+zu2+as.vector(anss2$b)
# cor(u[vv2,], DT$GY[vv2])
```

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
GT <- GT_cpdata
MP <- MP_cpdata
### mimic two fields
A <- A.mat(GT)
mix <- mmer(Yield~1,
            random=~vsr(id, Gu=A) +
              vsr(Rowf) +
              vsr(Colf) +
              spl2Da(Row,Col), nIters=3,
            rcov=~vsr(units),
```

```
            data=DT, verbose = FALSE)
summary(mix)
```

```
## ============================================================
##          Multivariate Linear Mixed Model fit by REML
## **********************  sommer 4.3  **********************
## ============================================================
##          logLik      AIC     BIC Method Converge
## Value -151.2647 304.5293 308.421     NR    FALSE
## ============================================================
## Variance-Covariance components:
##                      VarComp VarCompSE Zratio Constraint
## u:id.Yield-Yield       792.6     317.6 2.4959   Positive
## u:Rowf.Yield-Yield     807.6     371.3 2.1750   Positive
## u:Colf.Yield-Yield     183.2     139.7 1.3121   Positive
## A:all.Yield-Yield      515.8     701.3 0.7354   Positive
## u:units.Yield-Yield   2918.4     292.8 9.9667   Positive
## ============================================================
## Fixed effects:
##   Trait      Effect Estimate Std.Error t.value
## 1 Yield (Intercept)    132.1     8.761   15.08
## ============================================================
## Groups and observations:
##        Yield
## u:id     363
## u:Rowf    13
## u:Colf    36
## A:all    168
## ============================================================
## Use the '$' sign to access results and parameters
```

```
# make a plot to observe the spatial effects found by the spl2D()
W <- with(DT,spl2Da(Row,Col)) # 2D spline incidence matrix
DT$spatial <- W$Z$`A:all`%*%mix$U$`A:all`$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 `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 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=~ vsr(id, Gu=A, Gtc=unsm(2))
#               + vsr(Rowf,Gtc=diag(2))
```

```
#                 + vsr(Colf,Gtc=diag(2)),
#                 rcov=~ vsr(units, Gtc=unsm(2)), nIters=3,
#                 data=DT, verbose = FALSE)
```

Now you can extract the BLUPs using `randef(ans.m)` or simply `ans.m$U`. Also, genetic correlations and heritabilities can be calculated easily.

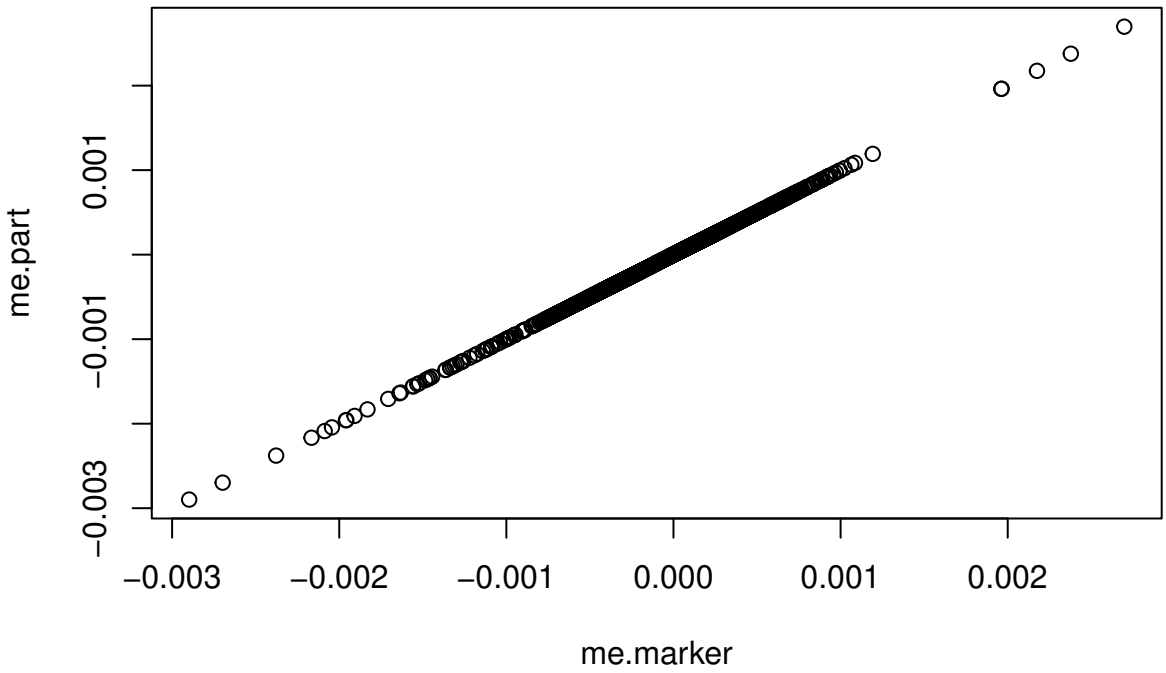
```
# cov2cor(ans.m$sigma$`u:id`)
```

## SECTION 3: 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:

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

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


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

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

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

mix.part <- mmer(color~1,
                 random=~Rowf+vsr(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, 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)
#
# modeld <- mmer(cbind(X1,X2) ~ UX - 1,
#                random = ~usr(id,Gu=D),
#                rcov = ~usr(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 = ~vsr(id,Gu=G),
#                rcov = ~vsr(units),
#                data=DTn, verbose = FALSE)
#
# ## compare regular and transformed blups
# plot(x=(solve(t(U)))%*%modeld$U$`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 the fact 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")
##############################
## 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
```

```
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, nIters=3,
            data=DT, verbose = FALSE)
vc <- summary(mix2)$varcomp; vc
```

```
##                                 VarComp  VarCompSE   Zratio Constraint
## femalef:malef:setf.yield-yield 0.0795986 0.07959526 1.000042   Positive
## malef:setf.yield-yield         0.3875096 0.27561291 1.405992   Positive
## units.yield-yield              0.1080557 0.05294578 2.040875   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
##  1.550038 -1.231644
```

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")
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
```

```
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])
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))
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
```

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

```
mix2 <- mmer(yield~ setf + setf:repf ,
             random=~femalef:malef:setf + malef:setf + femalef:setf,
             nIters=3,
             data=DT, verbose = FALSE)
vc <- summary(mix2)$varcomp; vc
```

```
##                                  VarComp VarCompSE    Zratio Constraint
## femalef:malef:setf.yield-yield  2235.754  2320.298 0.9635634   Positive
## malef:setf.yield-yield          5464.494  3484.990 1.5680085   Positive
## femalef:setf.yield-yield        2722.922  2316.795 1.1752969   Positive
## units.yield-yield              18569.305  2665.240 6.9672171   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
## 10891.689  8943.017
```

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
#### create the variance-covariance matrix
A <- A.mat(GT) # additive relationship matrix
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)
```

First we fit a regular GWAS/EMMAX using the GWAS function available in sommer that first calculates variance components and then fits a regression marker by marker as a fixed effect.

```
############################
#### Regular GWAS/EMMAX approach
############################
mix2 <- GWAS(color~1,
             random=~vsr(id, Gu=A) + Rowf + Colf,
             rcov=~units, M=GT, gTerm = "u:id",
             verbose = FALSE, nIters=3,
             data=DT)
```

## Performing GWAS evaluation

To compare EMMAX to the approach proposed by Gualdron-Duarte et al. (2014) and Bernal-Rubio et al. (2016) we will start fitting an RRBLUP/SNP-BLUP model to show that the estimate of the marker effect by its standard error is mathematically equivalent to fixed regression EMMAX GWAS.

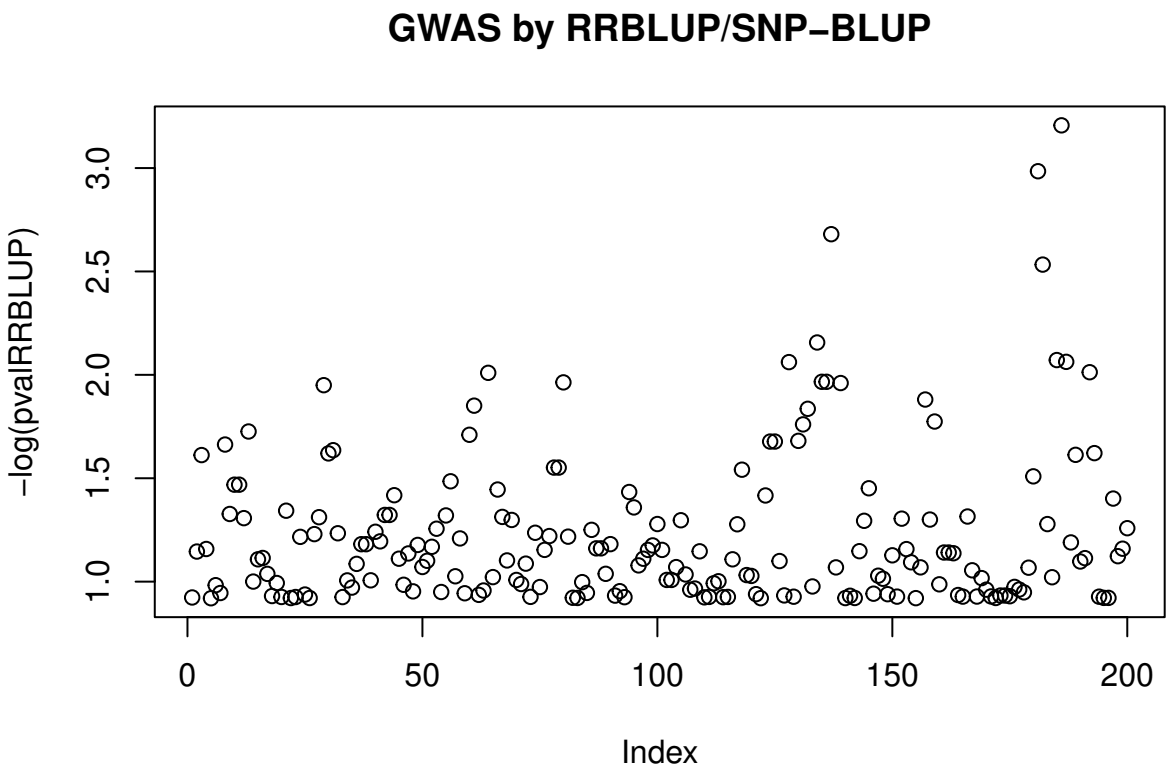
```
############################
#### GWAS by RRBLUP approach
############################
Z <- GT[as.character(DT$id),]
mixRRBLUP <- mmer(color~1,
                  random=~vsr(Z) + Rowf + Colf,
                  rcov=~units, nIters=3,
                  verbose = FALSE,
                  data=DT)

a <- mixRRBLUP$U$`u:Z`$color # marker effects
se.a <- sqrt(diag(kronecker(diag(ncol(Z)),mixRRBLUP$sigma$`u:Z`) - mixRRBLUP$PevU$`u:Z`$color)) # SE of
t.stat <- a/se.a # t-statistic
pvalRRBLUP <- dt(t.stat,df=n-k-1) # -log10(pval)
```

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.

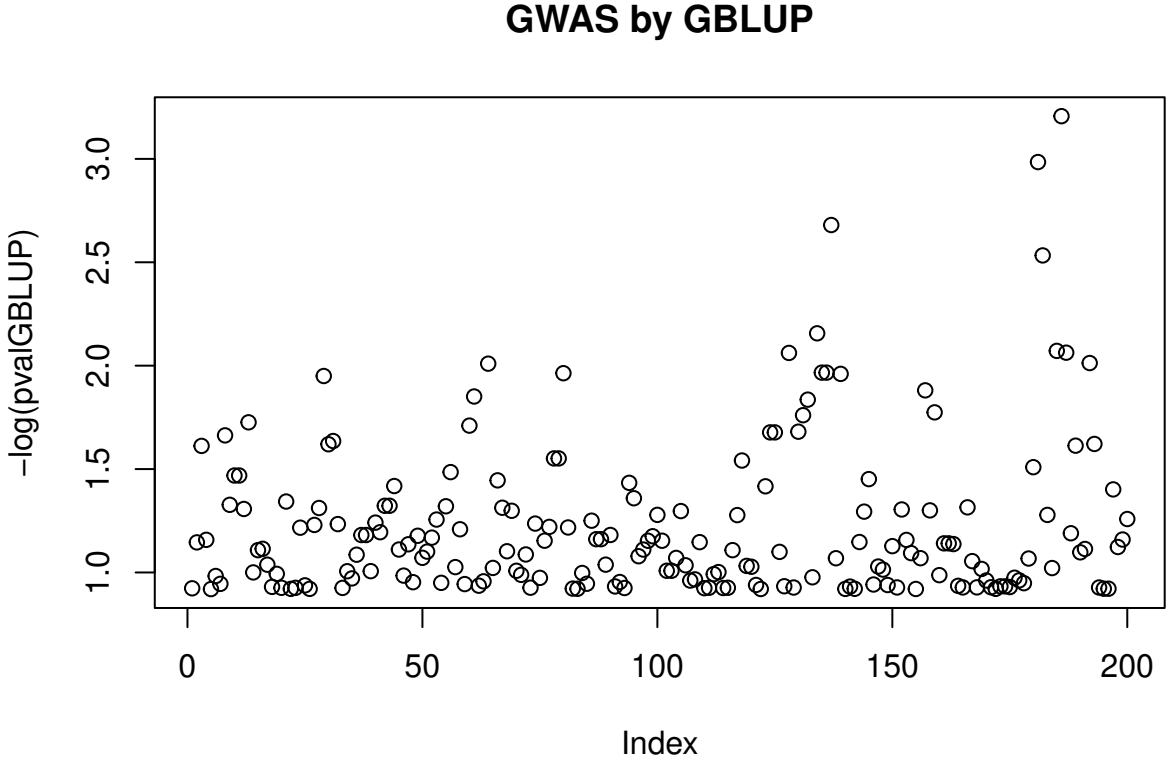
```
############################
#### GWAS by GBLUP approach
############################
M<- GT
MMT <-tcrossprod(M) ## MM' = additive relationship matrix
MMTinv<-solve(MMT + diag(1e-6, ncol(MMT), ncol(MMT))) ## inverse of MM'
MTMMTinv<-t(M)%*%MMTinv # M' %*% (M'M)-
mixGBLUP <- mmer(color~1,
                 random=~vsr(id, Gu=MMT) + Rowf + Colf,
                 rcov=~units, nIters=3,
                 verbose = FALSE,
                 data=DT)
a.from.g <-MTMMTinv%*%matrix(mixGBLUP$U$`u:id`$color,ncol=1)
var.g <- kronecker(MMT,mixGBLUP$sigma$`u:id`) - mixGBLUP$PevU$`u:id`$color
var.a.from.g <- t(M)%*%MMTinv%*% (var.g) %*% t(MMTinv)%*%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.

```
############################
#### Compare results
############################
# plot(mix2$scores[,1], main="GWAS")
plot(-log(pvalRRBLUP), main="GWAS by RRBLUP/SNP-BLUP")
```




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



**Final remarks**

Keep in mind that mmer uses a direct inversion (DI) algorithm which can be very slow for large datasets with many records. When datasets have more records than coefficients to be estimated please shift to the use of the mmec function.

## 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.