

Quick start for the sommer package

Giovanny Covarrubias-Pazaran

2019-01-03

The sommer package was developed to provide R users a powerful and reliable multivariate mixed model solver. The package is focused in problems of the type $p > n$ (more effects to estimate than observations) and its core algorithm is coded in C++ using the Armadillo library. This package allows the user to fit mixed models with the advantage of specifying the variance-covariance structure for the random effects, and specify heterogeneous variances, and obtain other parameters such as BLUPs, BLUEs, residuals, fitted values, variances for fixed and random effects, etc.

The purpose of this quick start guide is to show the flexibility of the package under certain common scenarios:

B1) Background on mixed models

B2) Background on covariance structures

- 1) Univariate homogeneous variance models
- 2) Univariate heterogeneous variance models
- 3) Univariate unstructured variance models
- 4) Multivariate homogeneous variance models
- 5) Multivariate heterogeneous variance models
- 6) Multivariate unstructured variance models
- 7) Details on special functions for variance models
 - the major `vs()` function for special variance models and its auxiliars:
 - `at()` specific levels heterogeneous variance structure
 - `ds()` diagonal covariance structure
 - `us()` unstructured covariance
 - `cs()` customized covariance structure
- 8) The specification of constraints in the variance components (Gtc argument)
 - `unsm()` unstructured constraint
 - `uncm()` unconstrained
 - `fixm()` fixed constraint
 - `fcm()` constraints on fixed effects
- 9) Special functions for special models
 - Random regression models
 - Overlayed models
 - Spatial models
 - GWAS models
 - Customized random effects
- 10) Final remarks

B1) Background on mixed models

The core of the package is 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 algorithms (Tunnicliffe 1989; Gilmour et al. 1995; Lee et al. 2016). From 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, 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:

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

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

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

$$\mathbf{X} = \begin{bmatrix} X_1 & 0 & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 & X_t \end{bmatrix}$$

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

where K is the relationship or covariance matrix for the k th random effect ($u=1, \dots, k$), and $H=I$ is an identity matrix or a partial 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.

B2) Background on covariance structures

One of the major strenghts of linear mixed models is the flexibility to specify variance-covariance structures at all levels. In general, variance structures of mixed models can be seen as tensor (kronecker) products of multiple variance-covariance stuctures. For example, a multi-response model (i.e. 2 traits) where “g” individuals (i.e. 100 individuals) are tested in “e” treatments (i.e. 3 environments), the variance-covariance for the random effect “individuals” can be seen as the following multiplicative model:

$$\mathbf{T} \otimes \mathbf{G} \otimes \mathbf{A}$$

where:

$$\mathbf{T} = \begin{bmatrix} \sigma_{g_{t1,t1}}^2 & \sigma_{g_{t1,t2}} \\ \sigma_{g_{t2,t1}} & \sigma_{g_{t2,t2}}^2 \end{bmatrix}$$

is the covariance structure for individuals among traits.

$$\mathbf{G} = \begin{bmatrix} \sigma_{g_{e1,e1}}^2 & \sigma_{g_{e1,e2}} & \sigma_{g_{e1,e3}} \\ \sigma_{g_{e2,e1}} & \sigma_{g_{e2,e2}}^2 & \sigma_{g_{e2,e3}} \\ \sigma_{g_{e3,e1}} & \sigma_{g_{e3,e2}} & \sigma_{g_{e3,e3}}^2 \end{bmatrix}$$

is the covariance structure for individuals among environments.

and \mathbf{A} is a square matrix representing the covariance among the levels of the individuals (any known relationship matrix).

The \mathbf{T} and \mathbf{G} covariance structures shown above are unknown matrices to be estimated whereas \mathbf{A} is known. The \mathbf{T} and \mathbf{G} matrices shown above are called as unstructured (US) covariance matrices, although this type is just one example from several covariance structures that the linear mixed models enable. For example, other popular covariance structures are:

Diagonal (DIAG) covariance structures

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_{g_{e1,e1}}^2 & 0 & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 & \sigma_{g_{ei,ei}}^2 \end{bmatrix}$$

Compound simmetry (CS) covariance structures

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_g^2 + \sigma_{ge}^2 & \sigma_g^2 & \sigma_g^2 & \sigma_g^2 \\ \sigma_g^2 & \sigma_g^2 + \sigma_{ge}^2 & \sigma_g^2 & \sigma_g^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_g^2 & \sigma_g^2 & \sigma_g^2 & \sigma_g^2 + \sigma_{ge}^2 \end{bmatrix}$$

First order autoregressive (AR1) covariance structures

$$\mathbf{\Sigma} = \sigma^2 \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 \\ \rho & 1 & \rho & \rho^2 \\ \rho^2 & \rho & 1 & \rho \\ \rho^3 & \rho^2 & \rho & 1 \end{bmatrix}$$

or the already mentioned Unstructured (US) covariance structures

$$\Sigma = \begin{bmatrix} \sigma_{g_{e1,e1}}^2 & \sigma_{g_{e1,e2}} & \sigma_{g_{e1,e3}} \\ \vdots & \ddots & \vdots \\ \sigma_{g_{e3,e1}} & \sigma_{g_{e3,e2}} & \sigma_{g_{e3,e3}}^2 \end{bmatrix}$$

among others. Sommer has the capabilities to fit some of these covariance structures in the mixed model machinery.

1) Univariate homogeneous variance models

This type of models refer to single response models where a variable of interest (i.e. genotypes) needs to be analyzed as interacting with a 2nd random effect (i.e. environments), but you assume that across environments the genotypes have the same variance component. This is the so-called compound symmetry (CS) model.

```
library(sommer)
data(DT_example)
head(DT)
```

```
##           Name      Env Loc Year      Block Yield      Weight
## 33  Manistee(MSL292-A) CA.2013  CA 2013  CA.2013.1      4 -1.904711
## 65           CO02024-9W CA.2013  CA 2013  CA.2013.1      5 -1.446958
## 66  Manistee(MSL292-A) CA.2013  CA 2013  CA.2013.2      5 -1.516271
## 67           MSL007-B CA.2011  CA 2011  CA.2011.2      5 -1.435510
## 68           MSR169-8Y CA.2013  CA 2013  CA.2013.1      5 -1.469051
## 103          AC05153-1W CA.2013  CA 2013  CA.2013.1      6 -1.307167
```

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

```
## iteration      LogLik      wall      cpu(sec)      restrained
##      1      -31.2668  20:45:43      0      0
##      2      -23.2804  20:45:43      0      0
##      3      -20.4746  20:45:43      0      0
##      4      -20.1501  20:45:43      0      0
##      5      -20.1454  20:45:43      0      0
##      6      -20.1454  20:45:43      0      0
```

```
summary(ans1)
```

```
## =====
##           Multivariate Linear Mixed Model fit by REML
## ***** sommer 3.8 *****
## =====
##           logLik      AIC      BIC Method Converge
## Value -20.14538 46.29075 55.95182      NR      TRUE
## =====
## Variance-Covariance components:
##           VarComp VarCompSE Zratio Constraint
## Name.Yield-Yield      3.682      1.691  2.177      Positive
## Env:Name.Yield-Yield      5.173      1.495  3.460      Positive
## units.Yield-Yield      4.366      0.647  6.748      Positive
## =====
## Fixed effects:
```

```
## Trait      Effect Estimate Std.Error t.value
## 1 Yield (Intercept) 16.496 0.6855 24.065
## 2 Yield EnvCA.2012 -5.777 0.7558 -7.643
## 3 Yield EnvCA.2013 -6.380 0.7960 -8.015
## =====
## Groups and observations:
##      Yield
## Name      41
## Env:Name 123
## =====
## Use the '$' sign to access results and parameters
```

2) Univariate heterogeneous variance models

Very often in multi-environment trials, the assumption that the genetic variance or the residual 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. This requires a CS+DIAG model (also called heterogeneous CS model).

```
data(DT_example)
head(DT)
```

```
##      Name      Env Loc Year      Block Yield      Weight
## 33  Manistee(MSL292-A) CA.2013  CA 2013  CA.2013.1      4 -1.904711
## 65      C002024-9W CA.2013  CA 2013  CA.2013.1      5 -1.446958
## 66  Manistee(MSL292-A) CA.2013  CA 2013  CA.2013.2      5 -1.516271
## 67      MSL007-B CA.2011  CA 2011  CA.2011.2      5 -1.435510
## 68      MSR169-8Y CA.2013  CA 2013  CA.2013.1      5 -1.469051
## 103     AC05153-1W CA.2013  CA 2013  CA.2013.1      6 -1.307167
```

```
ans2 <- mmmer(Yield~Env,
              random= ~Name + vs(ds(Env),Name),
              rcov= ~ vs(ds(Env),units),
              data=DT)
```

```
## iteration    LogLik      wall    cpu(sec)    restrained
##      1      -31.2668  20:45:43         0           0
##      2      -19.8549  20:45:43         0           0
##      3      -15.9797  20:45:43         0           0
##      4      -15.4374  20:45:44         1           0
##      5      -15.43  20:45:44         1           0
##      6      -15.4298  20:45:44         1           0
```

```
summary(ans2)
```

```
## =====
##      Multivariate Linear Mixed Model fit by REML
## ***** sommer 3.8 *****
## =====
##      logLik      AIC      BIC Method Converge
## Value -15.42983 36.85965 46.52072      NR      TRUE
## =====
## Variance-Covariance components:
##      VarComp VarCompSE Zratio Constraint
## Name.Yield-Yield      2.963      1.496 1.980 Positive
```

```
## CA.2011:Name.Yield-Yield    10.146    4.507  2.251  Positive
## CA.2012:Name.Yield-Yield     1.878    1.870  1.004  Positive
## CA.2013:Name.Yield-Yield     6.629    2.503  2.649  Positive
## CA.2011:units.Yield-Yield    4.942    1.525  3.242  Positive
## CA.2012:units.Yield-Yield    5.725    1.312  4.363  Positive
## CA.2013:units.Yield-Yield    2.560    0.640  4.000  Positive
## =====
## Fixed effects:
##   Trait      Effect Estimate Std.Error t.value
## 1 Yield (Intercept)  16.508    0.8268  19.965
## 2 Yield  EnvCA.2012   -5.817    0.8575  -6.783
## 3 Yield  EnvCA.2013   -6.412    0.9356  -6.854
## =====
## Groups and observations:
##           Yield
## Name          41
## CA.2011:Name    41
## CA.2012:Name    41
## CA.2013:Name    41
## =====
## Use the '$' sign to access results and parameters
```

As you can see the special function `at` or `diag` can be used to indicate that there's a different variance for the genotypes in each environment. Same was done for the residual. The difference between `at` and `diag` is that the `at` function can be used to specify the levels or specific environments where the variance is different.

3) Unstructured variance models

A more relaxed assumption than the CS+DIAG model is the unstructured model (US) which assumes that among the levels of certain factor (i.e. Environments) there's a covariance structure of a second random effect (i.e. Genotypes). This can be done in sommer using the `us(.)` function:

```
data(DT_example)
head(DT)
```

```
##           Name      Env Loc Year      Block Yield      Weight
## 33  Manistee(MSL292-A) CA.2013  CA 2013  CA.2013.1      4 -1.904711
## 65           CO02024-9W CA.2013  CA 2013  CA.2013.1      5 -1.446958
## 66  Manistee(MSL292-A) CA.2013  CA 2013  CA.2013.2      5 -1.516271
## 67           MSL007-B CA.2011  CA 2011  CA.2011.2      5 -1.435510
## 68           MSR169-8Y CA.2013  CA 2013  CA.2013.1      5 -1.469051
## 103          AC05153-1W CA.2013  CA 2013  CA.2013.1      6 -1.307167
```

```
ans3 <- mmmer(Yield~Env,
              random=~vs(us(Env),Name),
              rcov=~vs(us(Env),units),
              data=DT)
```

```
## iteration    LogLik      wall    cpu(sec)    restrained
##      1      -37.9059  20:45:44      0          0
##      2      -17.9745  20:45:44      0          0
##      3      -12.2427  20:45:44      0          0
##      4      -11.5121  20:45:44      0          0
##      5      -11.5001  20:45:44      0          0
##      6      -11.4997  20:45:44      0          0
```

```
summary(ans3)
```

```
## =====
##           Multivariate Linear Mixed Model fit by REML
## ***** sommer 3.8 *****
## =====
##           logLik      AIC      BIC Method Converge
## Value -11.49971 28.99943 38.66049      NR      TRUE
## =====
## Variance-Covariance components:
##                               VarComp VarCompSE      Zratio Constraint
## CA.2011:Name.Yield-Yield      15.665 5.421e+00 2.890e+00   Positive
## CA.2012:CA.2011:Name.Yield-Yield 6.110 2.485e+00 2.459e+00   Unconstr
## CA.2012:Name.Yield-Yield      4.530 1.821e+00 2.488e+00   Positive
## CA.2013:CA.2011:Name.Yield-Yield 6.384 3.066e+00 2.082e+00   Unconstr
## CA.2013:CA.2012:Name.Yield-Yield 0.393 1.523e+00 2.580e-01   Unconstr
## CA.2013:Name.Yield-Yield      8.597 2.484e+00 3.461e+00   Positive
## CA.2011:units.Yield-Yield      4.970 1.532e+00 3.243e+00   Positive
## CA.2012:CA.2011:units.Yield-Yield 4.087 2.436e-16 1.678e+16   Unconstr
## CA.2012:units.Yield-Yield      5.673 1.301e+00 4.361e+00   Positive
## CA.2013:CA.2011:units.Yield-Yield 4.087 0.000e+00      Inf   Unconstr
## CA.2013:CA.2012:units.Yield-Yield 4.087 0.000e+00      Inf   Unconstr
## CA.2013:units.Yield-Yield      2.557 6.393e-01 4.000e+00   Positive
## =====
## Fixed effects:
##   Trait      Effect Estimate Std.Error t.value
## 1 Yield (Intercept) 16.331      0.8137 20.070
## 2 Yield  EnvCA.2012  -5.696      0.7404 -7.693
## 3 Yield  EnvCA.2013  -6.271      0.8191 -7.656
## =====
## Groups and observations:
##                               Yield
## CA.2011:Name                41
## CA.2012:CA.2011:Name        82
## CA.2012:Name                 41
## CA.2013:CA.2011:Name        82
## CA.2013:CA.2012:Name        82
## CA.2013:Name                 41
## =====
## Use the '$' sign to access results and parameters
```

As can be seen the `us(Env)` indicates that the genotypes (Name) can have a covariance structure among environments (Env).

4) Multivariate homogeneous variance models

Currently there's a great push for multi-response models. This is motivated by the correlation that certain variables hide and that could benefit in the prediction perspective. In `sommer` to specify multivariate models the response requires the use of the `cbind()` function in the response, and the `us(trait)`, `diag(trait)`, or `at(trait)` functions in the random part of the model.

```
data(DT_example)
head(DT)
```

```
##           Name      Env Loc Year      Block Yield      Weight
## 33  Manistee(MSL292-A) CA.2013  CA 2013  CA.2013.1      4 -1.904711
## 65           C002024-9W CA.2013  CA 2013  CA.2013.1      5 -1.446958
## 66  Manistee(MSL292-A) CA.2013  CA 2013  CA.2013.2      5 -1.516271
## 67           MSL007-B CA.2011  CA 2011  CA.2011.2      5 -1.435510
## 68           MSR169-8Y CA.2013  CA 2013  CA.2013.1      5 -1.469051
## 103          AC05153-1W CA.2013  CA 2013  CA.2013.1      6 -1.307167
```

```
DT$EnvName <- paste(DT$Env,DT$Name)
ans4 <- mmer(cbind(Yield, Weight) ~ Env,
             random= ~ vs(Name) + vs(EnvName),
             rcov= ~ vs(units),
             data=DT)
```

```
## iteration      LogLik      wall      cpu(sec)      restrained
##      1      66.0395  20:45:45      0      0
##      2     131.529  20:45:46      1      0
##      3     162.769  20:45:46      1      0
##      4     166.983  20:45:47      2      0
##      5     167.025  20:45:47      2      0
##      6     167.025  20:45:48      3      0
```

```
summary(ans4)
```

```
## =====
##           Multivariate Linear Mixed Model fit by REML
## ***** sommer 3.8 *****
## =====
##           logLik      AIC      BIC Method Converge
## Value 167.0252 -322.0505 -298.5695      NR      TRUE
## =====
## Variance-Covariance components:
##           VarComp VarCompSE Zratio Constraint
## u:Name.Yield-Yield      3.7089      1.68117      2.206      Positive
## u:Name.Yield-Weight      0.9071      0.37944      2.391      Unconstr
## u:Name.Weight-Weight      0.2243      0.08775      2.557      Positive
## u:EnvName.Yield-Yield      5.0921      1.47879      3.443      Positive
## u:EnvName.Yield-Weight      1.0269      0.30767      3.338      Unconstr
## u:EnvName.Weight-Weight      0.2101      0.06661      3.154      Positive
## u:units.Yield-Yield      4.3837      0.64941      6.750      Positive
## u:units.Yield-Weight      0.9077      0.14145      6.417      Unconstr
## u:units.Weight-Weight      0.2280      0.03377      6.751      Positive
## =====
## Fixed effects:
##           Trait      Effect Estimate Std.Error t.value
## 1 Yield (Intercept)      16.4093      0.6783      24.191
## 2 Weight (Intercept)      0.9806      0.1497      6.550
## 3 Yield EnvCA.2012      -5.6844      0.7474      -7.606
## 4 Weight EnvCA.2012      -1.1846      0.1593      -7.439
## 5 Yield EnvCA.2013      -6.2952      0.7850      -8.019
## 6 Weight EnvCA.2013      -1.3559      0.1681      -8.065
## =====
## Groups and observations:
##           Yield Weight
## u:Name      41      41
```



```
## u:EnvName      94      94
## =====
## Use the '$' sign to access results and parameters
```

You may notice that we have added the `us(trait)` behind the random effects. This is to indicate the structure that should be assumed in the multivariate model. The `diag(trait)` used behind a random effect (i.e. Name) indicates that for the traits modeled (Yield and Weight) there's no a covariance component and should not be estimated, whereas `us(trait)` assumes that for such random effect, there's a covariance component to be estimated (i.e. covariance between Yield and Weight for the random effect Name). Same applies for the residual part (`rcov`).

5) Multivariate heterogeneous variance models

This is just an extension of the univariate heterogeneous variance models but at the multivariate level. This would be a CS+DIAG multivariate model:

```
data(DT_example)
head(DT)
```

```
##              Name      Env Loc Year      Block Yield      Weight
## 33  Manistee(MSL292-A) CA.2013  CA 2013  CA.2013.1      4 -1.904711
## 65              CO02024-9W CA.2013  CA 2013  CA.2013.1      5 -1.446958
## 66  Manistee(MSL292-A) CA.2013  CA 2013  CA.2013.2      5 -1.516271
## 67              MSL007-B CA.2011  CA 2011  CA.2011.2      5 -1.435510
## 68              MSR169-8Y CA.2013  CA 2013  CA.2013.1      5 -1.469051
## 103             AC05153-1W CA.2013  CA 2013  CA.2013.1      6 -1.307167
```

```
DT$EnvName <- paste(DT$Env,DT$Name)
ans5 <- mmer(cbind(Yield, Weight) ~ Env,
             random= ~ vs(Name) + vs(ds(Env),Name),
             rcov= ~ vs(ds(Env),units),
             data=DT)
```

```
## iteration      LogLik      wall      cpu(sec)      restrained
##      1      66.0395  20:45:49      1      0
##      2     138.617  20:45:50      2      0
##      3     172.682  20:45:51      3      0
##      4     177.662  20:45:52      4      0
##      5     177.801  20:45:54      6      0
##      6     177.813  20:45:55      7      0
##      7     177.815  20:45:56      8      0
##      8     177.815  20:45:57      9      0
```

```
summary(ans5)
```

```
## =====
##              Multivariate Linear Mixed Model fit by REML
## ***** sommer 3.8 *****
## =====
##              logLik      AIC      BIC Method Converge
## Value 177.8154 -343.6308 -320.1497      NR      TRUE
## =====
## Variance-Covariance components:
##              VarComp VarCompSE Zratio Constraint
## u:Name.Yield-Yield      3.31936      1.45269 2.2850      Positive
## u:Name.Yield-Weight      0.79393      0.32621 2.4338      Unconstr
```

```
## u.Name.Weight-Weight      0.19085    0.07503 2.5438    Positive
## CA.2011.Name.Yield-Yield   8.70657    4.01470 2.1687    Positive
## CA.2011.Name.Yield-Weight  1.77892    0.83926 2.1196    Unconstr
## CA.2011.Name.Weight-Weight 0.35966    0.17903 2.0089    Positive
## CA.2012.Name.Yield-Yield   2.57109    1.94951 1.3188    Positive
## CA.2012.Name.Yield-Weight  0.33245    0.39840 0.8345    Unconstr
## CA.2012.Name.Weight-Weight 0.03842    0.08595 0.4470    Positive
## CA.2013.Name.Yield-Yield   5.46908    2.16307 2.5284    Positive
## CA.2013.Name.Yield-Weight  1.34713    0.50479 2.6687    Unconstr
## CA.2013.Name.Weight-Weight 0.32902    0.12208 2.6952    Positive
## CA.2011:units.Yield-Yield  4.93852    1.52318 3.2422    Positive
## CA.2011:units.Yield-Weight 0.99447    0.32150 3.0932    Unconstr
## CA.2011:units.Weight-Weight 0.23982    0.07394 3.2433    Positive
## CA.2012:units.Yield-Yield  5.73887    1.31533 4.3631    Positive
## CA.2012:units.Yield-Weight 1.28009    0.30157 4.2448    Unconstr
## CA.2012:units.Weight-Weight 0.31806    0.07286 4.3652    Positive
## CA.2013:units.Yield-Yield  2.56127    0.63993 4.0024    Positive
## CA.2013:units.Yield-Weight 0.44569    0.12645 3.5246    Unconstr
## CA.2013:units.Weight-Weight 0.12232    0.03057 4.0009    Positive
## =====
## Fixed effects:
##      Trait      Effect Estimate Std.Error t.value
## 1  Yield (Intercept)  16.4243    0.7891  20.815
## 2  Weight (Intercept)   0.9866    0.1683   5.863
## 3  Yield  EnvCA.2012  -5.7339    0.8266  -6.937
## 4  Weight  EnvCA.2012  -1.1998    0.1698  -7.066
## 5  Yield  EnvCA.2013  -6.3128    0.8757  -7.209
## 6  Weight  EnvCA.2013  -1.3621    0.1915  -7.114
## =====
## Groups and observations:
##      Yield Weight
## u.Name      41    41
## CA.2011.Name 41    41
## CA.2012.Name 41    41
## CA.2013.Name 41    41
## =====
## Use the '$' sign to access results and parameters
```

6) Multivariate unstructured variance models

This is just an extension of the univariate unstructured variance models but at the multivariate level. This would be a US multivariate model:

```
data(DT_example)
head(DT)
```

```
##      Name      Env Loc Year      Block Yield      Weight
## 33  Manistee(MSL292-A) CA.2013  CA 2013  CA.2013.1      4 -1.904711
## 65      CO02024-9W CA.2013  CA 2013  CA.2013.1      5 -1.446958
## 66  Manistee(MSL292-A) CA.2013  CA 2013  CA.2013.2      5 -1.516271
## 67      MSL007-B CA.2011  CA 2011  CA.2011.2      5 -1.435510
## 68      MSR169-8Y CA.2013  CA 2013  CA.2013.1      5 -1.469051
## 103     AC05153-1W CA.2013  CA 2013  CA.2013.1      6 -1.307167
```

```
DT$EnvName <- paste(DT$Env,DT$Name)
ans6 <- mmer(cbind(Yield, Weight) ~ Env,
             random= ~ vs(us(Env),Name),
             rcov= ~ vs(ds(Env),units),
             data=DT)
```

```
## iteration      LogLik      wall      cpu(sec)      restrained
##      1         53.5859    20:45:58         1           0
##      2        140.385    20:46:0         3           0
##      3        176.108    20:46:1         4           0
##      4        181.454    20:46:2         5           0
##      5        181.696    20:46:4         7           0
##      6        181.755    20:46:5         8           0
##      7        181.777    20:46:6         9           0
##      8        181.787    20:46:8        11           0
##      9        181.791    20:46:9        12           0
##     10        181.793    20:46:10       13           0
##     11        181.794    20:46:11       14           0
##     12        181.795    20:46:13       16           0
```

```
summary(ans6)
```

```
## =====
##                      Multivariate Linear Mixed Model fit by REML
## ***** sommer 3.8 *****
## =====
##           logLik          AIC          BIC Method Converge
## Value 181.7947 -351.5895 -328.1085      NR      TRUE
## =====
## Variance-Covariance components:
##
##           VarComp VarCompSE Zratio Constraint
## CA.2011:Name.Yield-Yield      15.6405   5.35495  2.921   Positive
## CA.2011:Name.Yield-Weight       3.3577   1.14603  2.930   Unconstr
## CA.2011:Name.Weight-Weight       0.7180   0.24867  2.887   Positive
## CA.2012:CA.2011:Name.Yield-Yield    6.5289   2.48598  2.626   Unconstr
## CA.2012:CA.2011:Name.Yield-Weight    1.3505   0.52388  2.578   Unconstr
## CA.2012:CA.2011:Name.Weight-Weight    0.2842   0.11259  2.524   Unconstr
## CA.2012:Name.Yield-Yield       4.7896   1.86200  2.572   Positive
## CA.2012:Name.Yield-Weight       0.8641   0.38382  2.251   Unconstr
## CA.2012:Name.Weight-Weight       0.1693   0.08355  2.027   Positive
## CA.2013:CA.2011:Name.Yield-Yield    5.9941   2.93825  2.040   Unconstr
## CA.2013:CA.2011:Name.Yield-Weight    1.4235   0.64974  2.191   Unconstr
## CA.2013:CA.2011:Name.Weight-Weight    0.3379   0.14681  2.302   Unconstr
## CA.2013:CA.2012:Name.Yield-Yield    2.0970   1.44043  1.456   Unconstr
## CA.2013:CA.2012:Name.Yield-Weight    0.5232   0.32355  1.617   Unconstr
## CA.2013:CA.2012:Name.Weight-Weight    0.1339   0.07571  1.769   Unconstr
## CA.2013:Name.Yield-Yield       8.6264   2.47808  3.481   Positive
## CA.2013:Name.Yield-Weight       2.1046   0.58737  3.583   Unconstr
## CA.2013:Name.Weight-Weight       0.5124   0.14279  3.588   Positive
## CA.2011:units.Yield-Yield       4.9516   1.52693  3.243   Positive
## CA.2011:units.Yield-Weight       0.9993   0.32286  3.095   Unconstr
## CA.2011:units.Weight-Weight       0.2411   0.07432  3.244   Positive
## CA.2012:units.Yield-Yield       5.7783   1.32398  4.364   Positive
## CA.2012:units.Yield-Weight       1.2912   0.30401  4.247   Unconstr
```

```
## CA.2012:units.Weight-Weight          0.3211    0.07354  4.367    Positive
## CA.2013:units.Yield-Yield            2.5567    0.63882  4.002    Positive
## CA.2013:units.Yield-Weight           0.4452    0.12631  3.524    Unconstr
## CA.2013:units.Weight-Weight           0.1223    0.03056  4.001    Positive
## =====
## Fixed effects:
##      Trait      Effect Estimate Std.Error t.value
## 1  Yield (Intercept)  16.3339    0.8252  19.793
## 2 Weight (Intercept)   0.9677    0.1770   5.467
## 3  Yield EnvCA.2012   -5.6635    0.7447  -7.605
## 4 Weight EnvCA.2012   -1.1855    0.1604  -7.391
## 5  Yield EnvCA.2013   -6.2152    0.8338  -7.454
## 6 Weight EnvCA.2013   -1.3406    0.1805  -7.426
## =====
## Groups and observations:
##              Yield Weight
## CA.2011:Name      41     41
## CA.2012:CA.2011:Name 82     82
## CA.2012:Name      41     41
## CA.2013:CA.2011:Name 82     82
## CA.2013:CA.2012:Name 82     82
## CA.2013:Name      41     41
## =====
## Use the '$' sign to access results and parameters
```

Any number of random effects can be specified with different structures.

7) Details on special functions for variance models

the major `vs()` function for special variance models and its auxiliars

The `sommer` function `vs()` allows to construct complex variance models that are passed to the `mmer()` function it constitutes one of the most important features of the `sommer` package. Its specification of the `vs()` function has the form:

```
random=~vs(..., Gu, Gt, Gtc)
```

The idea is that the `vs()` function reflects the special variance structure that each random effect could have in the matrix notation:

$$var(u) = T \otimes E \otimes \dots \otimes A$$

where the `...` argument in the `vs()` function is used to specify the kronecker products from all matrices that form the variance for the random effect, where the auxiliar function `ds()`, `us()`, `cs()`, `at()`, can be used to define such structure variance structure. The idea is that a variance model for a random effect `x` (i.e. individuals) might require a more flexible model than just:

```
random=~x
```

For example, if individuals are tested in different time-points and environment, we can assume a different variance and covariance components among the individuals in the different environment-timepoint combinations. An example of variance structure of the type:

$$var(u) = T \otimes E \otimes S \otimes A$$

would be specified in the `vs()` function as:

`random=~vs(us(e),us(s),x, Gu=A, Gtc=T)`

where the **e** would be a column vector in a data frame for the environments, **s** a column vector in the dataframe for the time points, **x** is the vector in the dataframe for the identifier of individuals, **A** is a known square variance covariance matrix among individuals (usually an identity matrix; default if not specified), and **T** is a square matrices with as many rows and columns as the number of traits that specifies the trait covariance structure.

The auxiliary function to build the variance models for the random effect are: + `ds()` diagonal covariance structure + `us()` unstructured covariance + `at()` specific levels heterogeneous variance structure + `cs()` customized covariance structure

ds() to specify a diagonal (DIAG) covariance structures

A diagonal covariance structure looks like this:

$$\Sigma = \begin{bmatrix} \sigma_{g_{e1,e1}}^2 & 0 & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 & \sigma_{g_{ei,ei}}^2 \end{bmatrix}$$

Considering an example for one random effect (*g*; indicating i.e. individuals) evaluated in different treatment levels (*e*; indicating i.e. the different treatments) the model would look like:

`random=~vs(ds(e),g)`

us() to specify an unstructured (US) covariance

A unstructured covariance looks like this:

$$\mathbf{G} = \begin{bmatrix} \sigma_{g_{e1,e1}}^2 & \sigma_{g_{e1,e2}} & \sigma_{g_{e1,e3}} \\ \sigma_{g_{e2,e1}} & \sigma_{g_{e2,e2}}^2 & \sigma_{g_{e2,e3}} \\ \sigma_{g_{e3,e1}} & \sigma_{g_{e3,e2}} & \sigma_{g_{e3,e3}}^2 \end{bmatrix}$$

Considering same example for one random effect (*g*; indicating i.e. individuals) evaluated in different treatment levels (*e*; indicating i.e. the different treatments) the model would look like:

`random=~vs(us(e),g)`

at() to specify a level-specific heterogeneous variance

A diagonal covariance structure for specific levels of the second random effect looks like this:

$$\Sigma = \begin{bmatrix} \sigma_{g_{e1,e1}}^2 & 0 & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 & \sigma_{g_{ei,ei}}^2 \end{bmatrix}$$

Considering same example for one random effect (*g*; indicating i.e. individuals) evaluated in different treatment levels (*e*; indicating i.e. the different treatments A,B,C) the model would look like:

```
random=~vs(at(e,c("A","B")),g)
```

where the variance component for g is only fitted at levels A and B.

cs() to specify a level-specific variance-covariance structure

A customized covariance structure for specific levels of the second random effect (variance and covariances) looks i.e. like this:

$$\Sigma = \begin{bmatrix} \sigma_{g_{e1,e1}}^2 & \sigma_{g_{e1,e2}} & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 & \sigma_{g_{ei,ei}}^2 \end{bmatrix}$$

Considering same example for one random effect (g; indicating i.e. individuals) evaluated in different treatment levels (e; indicating i.e. the different treatments A,B,C) the model would look like:

```
random=~vs(cs(e,mm),g)
```

where mm indicates which variance and covariance components are estimated for g.

8) The specification of constraints in the variance components (Gtc argument)

One of the major strengths of sommer is its extreme flexibility to specify variance-covariance structures in the multi-trait framework. Since sommer 3.7 this is easily achieved by the use of the `vs()` function and it's argument `Gtc`. The `Gtc` argument expects a matrix of constraints for the variance-covariance components for the random effect filled with numbers according to the following rules:

0: parameter not to be estimated 1: estimated and constrained to be positive 2: estimated and unconstrained 3: not to be estimated but fixed value provided in Gt

Some useful function to specify quickly the constraint matrices are `unsm()` for unstructured, `uncm` for unconstrained, `fixm()` for fixed constraint, and `fcm()` for fixed effect constraints.

Consider a multi-trait model with 4 traits (y1,...,y4) and 1 random effects (u) and 1 fixed effect (x)

```
fixed=cbind(y1,y2,y3,y4)~x
```

```
random= ~vs(u, Gtc=?)
```

The constraint for the 4 x 4 matrix of variance covariance components to be estimated can be an:

- a) unstructured (variance components have to be positive and covariances either positive or negative)
- ```
random= ~vs(u, Gtc=unsm(4))
```

```
unsm(4)
```

```
[,1] [,2] [,3] [,4]
[1,] 1 2 2 2
[2,] 2 1 2 2
[3,] 2 2 1 2
[4,] 2 2 2 1
```

- b) unconstrained (any component variance or covariance can be positive or negative)
- ```
random= ~vs(u, Gtc=uncm(4))
```

```
uncm(4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    2    2    2
## [2,]    2    2    2    2
## [3,]    2    2    2    2
## [4,]    2    2    2    2
```

c) fixed (variance or covariance components indicated with a 3 are considered fixed and values are provided in the Gt argument) `random= ~vs(u, Gtc=fixm(4), Gt=mm)`

```
fixm(4)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    3    3    3    3
## [2,]    3    3    3    3
## [3,]    3    3    3    3
## [4,]    3    3    3    3
```

where mm is a 4 x 4 matrix with initial values for the variance components.

d) constraints for fixed effects `fixed= cbind(y1,y2,y3,y4)~vs(x, Gtc=fcm(c(1,0,1,0)))`

```
fcm(c(1,0,1,0))
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    0
## [3,]    0    1
## [4,]    0    0
```

where 1's and 0's indicate the traits where the fixed effect will be estimated (1's) and where it won't (0's).

9) Special functions for special models

Random regression models

In order to fit random regression models the user can use the `leg()` function to fit Legendre polynomials. This can be combined with other special covariance structures such as `ds()`, `us()`, etc.

```
library(orthopolynom)
```

```
## Loading required package: polynom
```

```
data(DT_legendre)
head(DT)
```

```
##      SUBJECT X      Y Xf
## 1.1      1 1 -0.7432795  1
## 2.1      2 1 -0.6669945  1
## 3.1      3 1 -4.2802751  1
## 4.1      4 1  4.1092149  1
## 5.1      5 1 -3.0317213  1
## 6.1      6 1  1.3506577  1
```

```
mRR2<-mmer(Y~ 1 + Xf
           , random=~ vs(us(leg(X,1)),SUBJECT)
           , rcov=~vs(units)
           , data=DT)
```

```
## iteration      LogLik      wall      cpu(sec)      restrained
##      1      -145.279    20:46:13         0         0
##      2      -138.353    20:46:13         0         0
##      3      -136.403    20:46:14         1         0
##      4      -136.224    20:46:14         1         0
##      5      -136.222    20:46:14         1         0
##      6      -136.222    20:46:14         1         0
```

```
summary(mRR2)$varcomp
```

```
##              VarComp VarCompSE  Zratio Constraint
## leg0:SUBJECT.Y-Y      2.5782969 0.6717242 3.838326   Positive
## leg1:leg0:SUBJECT.Y-Y 0.4765431 0.2394975 1.989763   Unconstr
## leg1:SUBJECT.Y-Y      0.3497299 0.2183229 1.601893   Positive
## u:units.Y-Y          2.6912226 0.3825197 7.035513   Positive
```

Here, a numeric covariate X is used to explain the trajectory of the SUBJECT's and combined with an unstructured covariance matrix. The details can be found in the theory.

GWAS models

Although genome wide association studies can be conducted through a variety of approaches, the use of mixed models to find association between markers and phenotypes still one of the most popular approaches. Two of the most classical and popular approaches is to test marker by marker through mixed modeling (1 model by marker) to obtain the marker effect and an statistic reflecting the level of association usually provided as the $-\log_{10}$ p-value. The second most popular approach is to assume that the genetic variance component is similar for all markers and therefore the variance components are only estimated once (1 model for all markers) and use the inverse of the phenotypic variance matrix (V^{-1}) to test all markers in the generalized linear model $b = (XV - X) - XV - y$. This makes the GWAS much faster and efficient without major losses. Given the straight forward extension, sommer provides the GWAS function which can fit both type of approaches (be aware that these are 2 among many existant in the literature) in univariate and multivariate models, that way genetically correlated traits can be tested together to increase the power of detection.

Here we show a simple GWAS model for an univariate example.

```
data(DT_cpdata)
#### create the variance-covariance matrix
A <- A.mat(GT) # additive relationship matrix
#### look at the data and fit the model
head(DT,3)
```

```
##      id Row Col Year      color  Yield FruitAver Firmness Rowf Colf
## P003 P003   3   1 2014 0.10075269 154.67      41.93  588.917    3    1
## P004 P004   4   1 2014 0.13891940 186.77      58.79  640.031    4    1
## P005 P005   5   1 2014 0.08681502  80.21      48.16  671.523    5    1
```

```
head(MP,3)
```

```
##              Locus Position Chrom
## 1 scaffold_77830_839         0     1
## 2 scaffold_39187_895         0     1
## 3 scaffold_50439_2379        0     1
```

```
GT[1:3,1:4]
```

```
##      scaffold_50439_2381 scaffold_39344_153 uneak_3436043 uneak_2632033
```



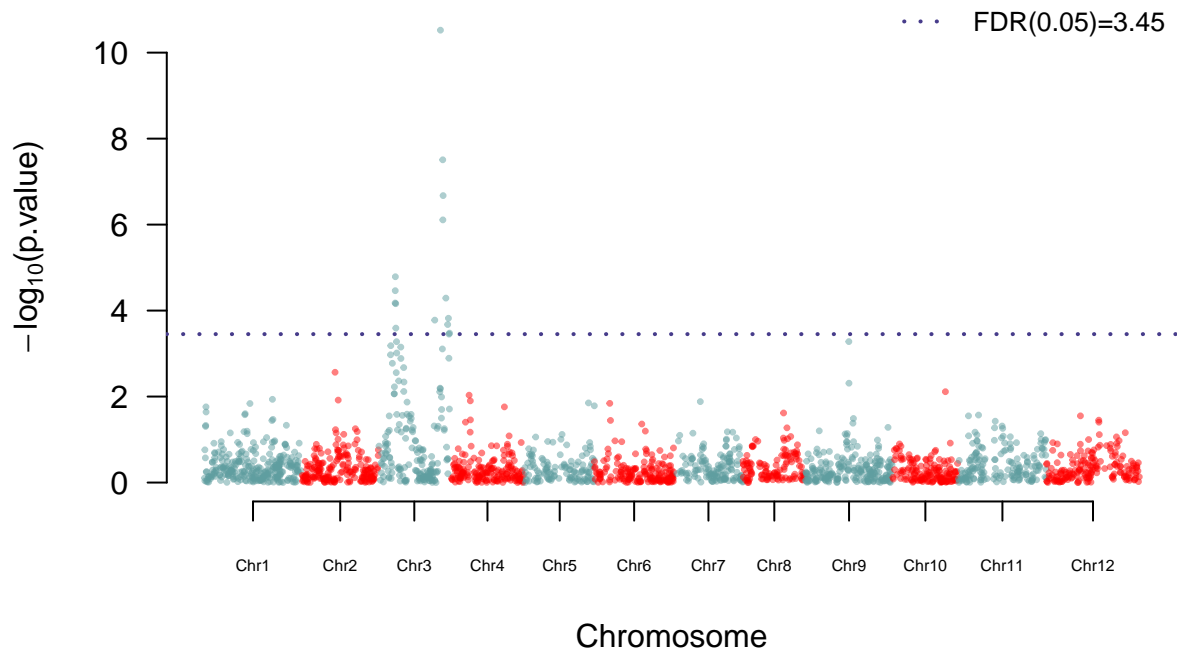
```
## P003          0          0          0          1
## P004          0          0          0          1
## P005          0         -1          0          1
```

```
mix1 <- GWAS(color~1,
  random=~vs(id,Gu=A)
  + Rowf + Colf,
  rcov=~units,
  data=DT,
  M=GT, gTerm = "u:id")
```

```
## iteration    LogLik      wall    cpu(sec)  restrained
##      1      -143.207  20:46:15         0         0
##      2      -117.977  20:46:16         1         0
##      3      -109.877  20:46:16         1         1
##      4      -108.178  20:46:16         1         1
##      5      -108.123  20:46:17         2         1
##      6      -108.12   20:46:17         2         1
##      7      -108.12   20:46:18         3         1
```

```
## Performing GWAS evaluation
```

```
ms <- as.data.frame(t(mix1$scores))
ms$Locus <- rownames(ms)
MP2 <- merge(MP,ms,by="Locus",all.x = TRUE);
manhattan(MP2, pch=20,cex=.5, PVCN = "color score")
```



Be aware that the marker matrix M has to be imputed (no missing data allowed) and make sure that the number of rows in the M matrix is equivalent to the levels of the gTerm specified (i.e. if the gTerm is “id” and has 300 levels or in other words 300 individuals, then M has dimensions 300 x p, being p the number of markers).

Overlaid models [the `overlay()` function]

Another very useful function is the `overlay` function, which allows to overlay matrices of different random effects and estimate a single variance component for the overlaid terms.

```
data("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   20:46:36      0         0
##      2      -6.487   20:46:36      0         0
##      3      -5.732   20:46:36      0         0
##      4      -5.67494  20:46:36      0         0
##      5      -5.67441  20:46:36      0         0
```

here the femalef and malef random effects are overlaid becoming a single random effect that has the same variance component.

Spatial models (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")
### 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   20:46:37         1           0
##      2      -152.064   20:46:37         1           0
##      3      -151.265   20:46:37         1           0
##      4      -151.202   20:46:38         2           0
##      5      -151.201   20:46:38         2           0
```

```
summary(mix)
```

```
## =====
##           Multivariate Linear Mixed Model fit by REML
## ***** sommer 3.8 *****
## =====
##           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
```

Notice that the job is done by the `spl2D()` function that takes the Row and Col information to fit a spatial kernel.

Customized random effects

One of the most powerful features of `sommer` is the ability to provide any customized matrix and estimate any random effect. For example:

```
data(DT_cpdata)
GT[1:4,1:4]
```

```
##      scaffold_50439_2381 scaffold_39344_153 uneak_3436043 uneak_2632033
## P003                0                0                0                1
## P004                0                0                0                1
## P005                0               -1                0                1
## P006               -1               -1               -1                0
```

```
#### look at the data and fit the model
```

```
mix1 <- mmer(Yield~1,
             random=~vs(list(GT)),
             rcov=~units,
             data=DT)
```

```
## iteration   LogLik      wall   cpu(sec)   restrained
##      1      -286.365  20:46:39         0         0
##      2      -236.78  20:46:40         1         0
##      3      -200.635  20:46:40         1         0
##      4      -180.045  20:46:40         1         0
##      5      -176.4   20:46:40         1         0
##      6      -176.211  20:46:41         2         0
##      7      -176.207  20:46:41         2         0
##      8      -176.207  20:46:41         2         0
```

the matrix GT is provided as a random effect by encapsulating the matrix in a list and provided in the `vs()` function.

10) 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.

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.
- Abdollahi Arpanahi R, Morota G, Valente BD, Kranis A, Rosa GJM, Gianola D. 2015. Assessment of bagging GBLUP for whole genome prediction of broiler chicken traits. *Journal of Animal Breeding and Genetics* 132:218-228.
- Tunncliffe W. 1989. On the use of marginal likelihood in time series model estimation. *JRSS* 51(1):15-27.