With field experiments, studying the correlation between the observed traits may not be an easy task. Indeed, in these experiments, subjects are not independent, but they are grouped by treatment factors (e.g., genotypes or weed control methods) or by blocking factors (e.g., blocks, plots, main-plots). I gave a solution based on traditional methods of data analyses.

In a recent paper, Piepho (2018) proposed a more advanced solution based on mixed models. He presented four examplary datasets and gave SAS code to analyse them, based on PROC MIXED. I was very interested in those examples, but I do not use SAS. Therefore, I tried to ‘transport’ the models in R, which turned out to be a difficult task. After struggling for awhile with several mixed model packages, I came to an acceptable solution, which I would like to share.

**A ‘routine’ experiment**

I will use the same example, which should allow us to compare the results with those obtained by using more traditional methods of data analyses. It is a genotype experiment, laid out in randomised complete blocks, with 27 wheat genotypes and three replicates. For each plot, the collegue who made the experiment recorded several traits, including yield (Yield) and weight of thousand kernels (TKW). The dataset ‘WheatQuality.csv’ is available on ‘gitHub’; it consists of 81 records (plots) and just as many couples of measures in all. The code below loads the necessary packages, the data and transforms the numeric variable ‘Block’ into a factor.

library(reshape2)

library(plyr)

library(nlme)

library(plyr)

library(asreml)

dataset <- read.csv("https://raw.githubusercontent.com/OnofriAndreaPG/aomisc/master/data/WheatQuality.csv", header=T)

dataset$Block <- factor(dataset$Block)

head(dataset)

## Genotype Block Height TKW Yield Whectol

## 1 arcobaleno 1 90 44.5 64.40 83.2

## 2 arcobaleno 2 90 42.8 60.58 82.2

## 3 arcobaleno 3 88 42.7 59.42 83.1

## 4 baio 1 80 40.6 51.93 81.8

## 5 baio 2 75 42.7 51.34 81.3

## 6 baio 3 76 41.1 47.78 81.1

I used the above dataset to calculate the Pearson’s correlation coefficient between Yield and TKW for:

1. plot measurements,
2. residuals,
3. treatment/block means.

Piepho (2018) showed that, for an experiment like this one, the above correlations can be estimated by coding a multiresponse mixed model, as follows:

\[ Y\_{ijk} = \mu\_i + \beta\_{ik} + \tau\_{ij} + \epsilon\_{ijk}\]

where \(Y\_{ijk}\) is the response for the trait \(i\), the rootstock \(j\) and the block \(k\), \(\mu\_i\) is the mean for the trait \(i\), \(\beta\_{ik}\) is the effect of the block \(k\) and trait \(i\), \(\tau\_{ij}\) is the effect of genotype \(j\) for the trait \(i\) and \(\epsilon\_{ijk}\) is the residual for each of 81 observations for two traits.

In the above model, the residuals \(\epsilon\_{ijk}\) need to be normally distributed and heteroscedastic, with trait-specific variances. Furthermore, residuals belonging to the same plot (the two observed traits) need to be correlated (correlation of errors).

Hans-Peter Piepho, in his paper, put forward the idea that the ‘genotype’ and ‘block’ effects for the two traits can be taken as random, even though they might be of fixed nature, especially the genotype effect. This idea makes sense, because, for this application, we are mainly interested in variances and covariances. Both random effects need to be heteroscedastic (trait-specific variance components) and there must be a correlation between the two traits.

To the best of my knowledge, there is no way to fit such a complex model with the ‘nlme’ package and related ‘lme()’ function (I’ll gave a hint later on, for a simpler model). Therefore, I decided to use the package ‘asreml’ (Butler et al., 2018), although this is not freeware. With the function ‘asreml()’, we need to specify the following components.

1. The response variables. When we set a bivariate model with ‘asreml’, we can ‘cbind()’ Yield and TKW and use the name ‘trait’ to refer to them.
2. The fixed model, that only contains the trait effect. The specification is, therefore, ‘cbind(Yield, TKW) ~ trait – 1’. Following Piepho (2018), I removed the intercept, to separately estimate the means for the two traits.
3. The random model, that is composed by the interactions ‘genotype x trait’ and ‘block x trait’. For both, I specified a general unstructured variance covariance matrix, so that the traits are heteroscedastic and correlated. Therefore, the random model is ~ Genotype:us(trait) + Block:us(trait).
4. The residual structure, where the observations in the same plot (the term ‘units’ is used in ‘asreml’ to represent the observational units, i.e. the plots) are heteroscedastic and correlated.

The model call is:

mod.asreml <- asreml(cbind(Yield, TKW) ~ trait - 1,

random = ~ Genotype:us(trait, init = c(3.7, -0.25, 1.7)) +

Block:us(trait, init = c(77, 38, 53)),

residual = ~ units:us(trait, init = c(6, 0.16, 4.5)),

data=dataset)

## Model fitted using the sigma parameterization.

## ASReml 4.1.0 Fri May 10 17:33:27 2019

## LogLik Sigma2 DF wall cpu

## 1 -641.556 1.0 160 17:33:27 0.0 (3 restrained)

## 2 -548.767 1.0 160 17:33:27 0.0

## 3 -448.970 1.0 160 17:33:27 0.0

## 4 -376.952 1.0 160 17:33:27 0.0

## 5 -334.100 1.0 160 17:33:27 0.0

## 6 -317.511 1.0 160 17:33:27 0.0

## 7 -312.242 1.0 160 17:33:27 0.0

## 8 -311.145 1.0 160 17:33:27 0.0

## 9 -311.057 1.0 160 17:33:27 0.0

## 10 -311.056 1.0 160 17:33:27 0.0

summary(mod.asreml)$varcomp[,1:3]

## component std.error z.ratio

## Block:trait!trait\_Yield:Yield 3.7104778 3.9364268 0.9426005

## Block:trait!trait\_TKW:Yield -0.2428390 1.9074544 -0.1273105

## Block:trait!trait\_TKW:TKW 1.6684568 1.8343662 0.9095549

## Genotype:trait!trait\_Yield:Yield 77.6346623 22.0956257 3.5135761

## Genotype:trait!trait\_TKW:Yield 38.8322972 15.0909109 2.5732242

## Genotype:trait!trait\_TKW:TKW 53.8616088 15.3520661 3.5084274

## units:trait!R 1.0000000 NA NA

## units:trait!trait\_Yield:Yield 6.0939037 1.1951128 5.0990195

## units:trait!trait\_TKW:Yield 0.1635551 0.7242690 0.2258209

## units:trait!trait\_TKW:TKW 4.4717901 0.8769902 5.0990195

The box above shows the results about the variance-covariance parameters. In order to get the correlations, I specified the necessary combinations of variance-covariance parameters. It is necessary to remember that estimates, in ‘asreml’, are named as V1, V2, … Vn, according to their ordering in model output.

parms <- mod.asreml$vparameters

vpredict(mod.asreml, rb ~ V2 / (sqrt(V1)\*sqrt(V3) ) )

## Estimate SE

## rb -0.09759916 0.7571335

vpredict(mod.asreml, rt ~ V5 / (sqrt(V4)\*sqrt(V6) ) )

## Estimate SE

## rt 0.6005174 0.130663

vpredict(mod.asreml, re ~ V9 / (sqrt(V8)\*sqrt(V10) ) )

## Estimate SE

## re 0.03133109 0.1385389

We see that the estimates are very close to those obtained by using the Pearson’s correlation coefficients (see my previous post). The advantage of this mixed model solution is that we can also test hypotheses in a relatively reliable way. For example, I tested the hypothesis that residuals are not correlated by:

1. coding a reduced model where residuals are heteroscedastic and independent, and
2. comparing this reduced model with the complete model by way of a REML-based Likelihood Ratio Test (LRT).

Removing the correlation of residuals is easily done, by changing the correlation structure from ‘us’ (unstructured variance-covariance matrix) to ‘idh’ (diagonal variance-covariance matrix).

mod.asreml2 <- asreml(cbind(Yield, TKW) ~ trait - 1,

random = ~ Genotype:us(trait) + Block:us(trait),

residual = ~ units:idh(trait),

data=dataset)

## Model fitted using the sigma parameterization.

## ASReml 4.1.0 Fri May 10 17:33:28 2019

## LogLik Sigma2 DF wall cpu

## 1 -398.023 1.0 160 17:33:28 0.0 (2 restrained)

## 2 -383.859 1.0 160 17:33:28 0.0

## 3 -344.687 1.0 160 17:33:28 0.0

## 4 -321.489 1.0 160 17:33:28 0.0

## 5 -312.488 1.0 160 17:33:28 0.0

## 6 -311.167 1.0 160 17:33:28 0.0

## 7 -311.083 1.0 160 17:33:28 0.0

## 8 -311.082 1.0 160 17:33:28 0.0

lrt.asreml(mod.asreml, mod.asreml2)

## Likelihood ratio test(s) assuming nested random models.

## (See Self & Liang, 1987)

##

## df LR-statistic Pr(Chisq)

## mod.asreml/mod.asreml2 1 0.05107 0.4106

Likewise, I tried to further reduce the model to test the significance of the correlation between block means and genotype means.

mod.asreml3 <- asreml(cbind(Yield, TKW) ~ trait - 1,

random = ~ Genotype:us(trait) + Block:idh(trait),

residual = ~ units:idh(trait),

data=dataset)

## Model fitted using the sigma parameterization.

## ASReml 4.1.0 Fri May 10 17:33:29 2019

## LogLik Sigma2 DF wall cpu

## 1 -398.027 1.0 160 17:33:29 0.0 (2 restrained)

## 2 -383.866 1.0 160 17:33:29 0.0

## 3 -344.694 1.0 160 17:33:29 0.0

## 4 -321.497 1.0 160 17:33:29 0.0

## 5 -312.496 1.0 160 17:33:29 0.0

## 6 -311.175 1.0 160 17:33:29 0.0

## 7 -311.090 1.0 160 17:33:29 0.0

## 8 -311.090 1.0 160 17:33:29 0.0

lrt.asreml(mod.asreml, mod.asreml3)

## Likelihood ratio test(s) assuming nested random models.

## (See Self & Liang, 1987)

##

## df LR-statistic Pr(Chisq)

## mod.asreml/mod.asreml3 2 0.066663 0.6399

mod.asreml4 <- asreml(cbind(Yield, TKW) ~ trait - 1,

random = ~ Genotype:idh(trait) + Block:idh(trait),

residual = ~ units:idh(trait),

data=dataset)

## Model fitted using the sigma parameterization.

## ASReml 4.1.0 Fri May 10 17:33:30 2019

## LogLik Sigma2 DF wall cpu

## 1 -406.458 1.0 160 17:33:30 0.0 (2 restrained)

## 2 -394.578 1.0 160 17:33:30 0.0

## 3 -352.769 1.0 160 17:33:30 0.0

## 4 -327.804 1.0 160 17:33:30 0.0

## 5 -318.007 1.0 160 17:33:30 0.0

## 6 -316.616 1.0 160 17:33:30 0.0

## 7 -316.549 1.0 160 17:33:30 0.0

## 8 -316.549 1.0 160 17:33:30 0.0

lrt.asreml(mod.asreml, mod.asreml4)

## Likelihood ratio test(s) assuming nested random models.

## (See Self & Liang, 1987)

##

## df LR-statistic Pr(Chisq)

## mod.asreml/mod.asreml4 3 10.986 0.003364 \*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

We see that only the genotype means are significantly correlated.

An alternative (and more useful) way to code the same model is by using the ‘corgh’ structure, instead of ‘us’. These two structures are totally similar, apart from the fact that the first one uses the correlations, instead of the covariances. Another difference, which we should consider when giving starting values, is that correlations come before variances.

mod.asreml.r <- asreml(cbind(Yield, TKW) ~ trait - 1,

random = ~ Genotype:corgh(trait, init=c(-0.1, 3.8, 1.8))

+ Block:corgh(trait, init = c(0.6, 77, 53)),

residual = ~ units:corgh(trait, init = c(0.03, 6, 4.5)),

data=dataset)

## Model fitted using the sigma parameterization.

## ASReml 4.1.0 Fri May 10 17:33:31 2019

## LogLik Sigma2 DF wall cpu

## 1 -632.445 1.0 160 17:33:31 0.0 (3 restrained)

## 2 -539.383 1.0 160 17:33:31 0.0 (2 restrained)

## 3 -468.810 1.0 160 17:33:31 0.0 (1 restrained)

## 4 -422.408 1.0 160 17:33:31 0.0

## 5 -371.304 1.0 160 17:33:31 0.0

## 6 -336.191 1.0 160 17:33:31 0.0

## 7 -317.547 1.0 160 17:33:31 0.0

## 8 -312.105 1.0 160 17:33:31 0.0

## 9 -311.118 1.0 160 17:33:31 0.0

## 10 -311.057 1.0 160 17:33:31 0.0

## 11 -311.056 1.0 160 17:33:31 0.0

summary(mod.asreml.r)$varcomp[,1:2]

## component std.error

## Block:trait!trait!TKW:!trait!Yield.cor -0.09759916 0.7571335

## Block:trait!trait\_Yield 3.71047783 3.9364268

## Block:trait!trait\_TKW 1.66845679 1.8343662

## Genotype:trait!trait!TKW:!trait!Yield.cor 0.60051740 0.1306748

## Genotype:trait!trait\_Yield 77.63466334 22.0981962

## Genotype:trait!trait\_TKW 53.86160957 15.3536792

## units:trait!R 1.00000000 NA

## units:trait!trait!TKW:!trait!Yield.cor 0.03133109 0.1385389

## units:trait!trait\_Yield 6.09390366 1.1951128

## units:trait!trait\_TKW 4.47179012 0.8769902

The advantage of this parameterisation is that we can test our hypotheses by easily setting up contraints on correlations. One way to do this is to run the model with the argument ‘start.values = T’. In this way I could derive a data frame (‘mod.init$parameters’), with the starting values for REML maximisation.

# Getting the starting values

mod.init <- asreml(cbind(Yield, TKW) ~ trait - 1,

random = ~ Genotype:corgh(trait, init=c(-0.1, 3.8, 1.8))

+ Block:corgh(trait, init = c(0.6, 77, 53)),

residual = ~ units:corgh(trait, init = c(0.03, 6, 4.5)),

data=dataset, start.values = T)

init <- mod.init$vparameters

init

## Component Value Constraint

## 1 Genotype:trait!trait!TKW:!trait!Yield.cor -0.10 U

## 2 Genotype:trait!trait\_Yield 3.80 P

## 3 Genotype:trait!trait\_TKW 1.80 P

## 4 Block:trait!trait!TKW:!trait!Yield.cor 0.60 U

## 5 Block:trait!trait\_Yield 77.00 P

## 6 Block:trait!trait\_TKW 53.00 P

## 7 units:trait!R 1.00 F

## 8 units:trait!trait!TKW:!trait!Yield.cor 0.03 U

## 9 units:trait!trait\_Yield 6.00 P

## 10 units:trait!trait\_TKW 4.50 P

We see that the ‘init’ data frame has three columns: (i) names of parameters, (ii) initial values and (iii) type of constraint (U: unconstrained, P = positive, F = fixed). Now, if we take the seventh row (correlation of residuals), set the initial value to 0 and set the third column to ‘F’ (meaning: keep the initial value fixed), we are ready to fit a model without correlation of residuals (same as the ‘model.asreml2’ above). What I had to do was just to pass this data frame as the starting value matrix for a new model fit (see the argument ‘R.param’, below).

init2 <- init

init2[8, 2] <- 0

init2[8, 3] <- "F"

mod.asreml.r2 <- asreml(cbind(Yield, TKW) ~ trait - 1,

random = ~ Genotype:corgh(trait)

+ Block:corgh(trait),

residual = ~ units:corgh(trait),

data=dataset, R.param = init2)

## Model fitted using the sigma parameterization.

## ASReml 4.1.0 Fri May 10 17:33:33 2019

## LogLik Sigma2 DF wall cpu

## 1 -1136.066 1.0 160 17:33:33 0.0 (1 restrained)

## 2 -939.365 1.0 160 17:33:33 0.0 (1 restrained)

## 3 -719.371 1.0 160 17:33:33 0.0 (1 restrained)

## 4 -550.513 1.0 160 17:33:33 0.0

## 5 -427.355 1.0 160 17:33:33 0.0

## 6 -353.105 1.0 160 17:33:33 0.0

## 7 -323.421 1.0 160 17:33:33 0.0

## 8 -313.616 1.0 160 17:33:33 0.0

## 9 -311.338 1.0 160 17:33:33 0.0

## 10 -311.087 1.0 160 17:33:33 0.0

## 11 -311.082 1.0 160 17:33:33 0.0

summary(mod.asreml.r2)$varcomp[,1:2]

## component std.error

## Block:trait!trait!TKW:!trait!Yield.cor -0.09516456 0.7572040

## Block:trait!trait\_Yield 3.71047783 3.9364268

## Block:trait!trait\_TKW 1.66845679 1.8343662

## Genotype:trait!trait!TKW:!trait!Yield.cor 0.60136047 0.1305180

## Genotype:trait!trait\_Yield 77.63463977 22.0809306

## Genotype:trait!trait\_TKW 53.86159936 15.3451466

## units:trait!R 1.00000000 NA

## units:trait!trait!TKW:!trait!Yield.cor 0.00000000 NA

## units:trait!trait\_Yield 6.09390366 1.1951128

## units:trait!trait\_TKW 4.47179012 0.8769902

lrt.asreml(mod.asreml.r2, mod.asreml.r)

## Likelihood ratio test(s) assuming nested random models.

## (See Self & Liang, 1987)

##

## df LR-statistic Pr(Chisq)

## mod.asreml.r/mod.asreml.r2 1 0.051075 0.4106

What is even more interesting is that ‘asreml’ permits to force the parameters to be linear combinations of one another. For instance, we can code a model where the residual correlation is contrained to be equal to the treatment correlation. To do so, we have to set up a two-column matrix (M), with row names matching the component names in the ‘asreml’ parameter vector. The matrix M should contain:

1. in the first column, the equality relationships (same number, same value)
2. in the second column, the coefficients for the multiplicative relationships

In this case, we would set the matrix M as follows:

firstCol <- c(1, 2, 3, 4, 5, 6, 7, 1, 8, 9)

secCol <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)

M <- cbind(firstCol, secCol)

dimnames(M)[[1]] <- mod.init$vparameters$Component

M

## firstCol secCol

## Genotype:trait!trait!TKW:!trait!Yield.cor 1 1

## Genotype:trait!trait\_Yield 2 1

## Genotype:trait!trait\_TKW 3 1

## Block:trait!trait!TKW:!trait!Yield.cor 4 1

## Block:trait!trait\_Yield 5 1

## Block:trait!trait\_TKW 6 1

## units:trait!R 7 1

## units:trait!trait!TKW:!trait!Yield.cor 1 1

## units:trait!trait\_Yield 8 1

## units:trait!trait\_TKW 9 1

Please note that in ‘firstCol’, the 1st and 8th element are both equal to 1, which contraints them to assume the same value. We can now pass the matrix M as the value of the ‘vcc’ argument in the model call.

mod.asreml.r3 <- asreml(cbind(Yield, TKW) ~ trait - 1,

random = ~ Genotype:corgh(trait)

+ Block:corgh(trait),

residual = ~ units:corgh(trait),

data=dataset, R.param = init, vcc = M)

## Model fitted using the sigma parameterization.

## ASReml 4.1.0 Fri May 10 17:33:34 2019

## LogLik Sigma2 DF wall cpu

## 1 -1122.762 1.0 160 17:33:34 0.0 (1 restrained)

## 2 -900.308 1.0 160 17:33:34 0.0

## 3 -665.864 1.0 160 17:33:34 0.0

## 4 -492.020 1.0 160 17:33:34 0.0

## 5 -383.085 1.0 160 17:33:34 0.0

## 6 -336.519 1.0 160 17:33:34 0.0 (1 restrained)

## 7 -319.561 1.0 160 17:33:34 0.0

## 8 -315.115 1.0 160 17:33:34 0.0

## 9 -314.540 1.0 160 17:33:34 0.0

## 10 -314.523 1.0 160 17:33:34 0.0

summary(mod.asreml.r3)$varcomp[,1:3]

## component std.error z.ratio

## Block:trait!trait!TKW:!trait!Yield.cor -0.1146908 0.7592678 -0.1510545

## Block:trait!trait\_Yield 3.6991785 3.9364622 0.9397216

## Block:trait!trait\_TKW 1.6601799 1.8344090 0.9050217

## Genotype:trait!trait!TKW:!trait!Yield.cor 0.2336876 0.1117699 2.0907919

## Genotype:trait!trait\_Yield 70.5970531 19.9293722 3.5423621

## Genotype:trait!trait\_TKW 48.9763800 13.8464106 3.5371174

## units:trait!R 1.0000000 NA NA

## units:trait!trait!TKW:!trait!Yield.cor 0.2336876 0.1117699 2.0907919

## units:trait!trait\_Yield 6.3989855 1.2811965 4.9945387

## units:trait!trait\_TKW 4.6952670 0.9400807 4.9945358

lrt.asreml(mod.asreml.r3, mod.asreml.r)

## Likelihood ratio test(s) assuming nested random models.

## (See Self & Liang, 1987)

##

## df LR-statistic Pr(Chisq)

## mod.asreml.r/mod.asreml.r3 1 6.9336 0.00423 \*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

From the output, we see that the residual and treatment correlations are equal in this latter model. We also see that this reduced model fits significantly worse than the complete model ‘mod.asreml.r’.

**Going freeware**

Considering that the block means are not correlated, if we were willing to take the ‘block’ effect as fixed, we could fit the resulting model also with the ‘nlme’ package and the function ‘lme()’ (Pinheiro and Bates, 2000). However, we should cast the model as a univariate model.

To this aim, the two variables (Yield and TKW) need to be piled up and a new factor (‘Trait’) needs to be introduced to identify the observations for the two traits. Another factor is also necessary to identify the different plots, i.e. the observational units. To perform such a restructuring, I used the ‘melt()’ function in the ‘reshape2’ package Wickham, 2007) and assigned the name ‘Y’ to the response variable, that is now composed by the two original variables Yield and TKW, one on top of the other.

dataset$Plot <- 1:81

mdataset <- melt(dataset[,c(-3,-6)], variable.name= "Trait", value.name="Y", id=c("Genotype", "Block", "Plot"))

mdataset$Block <- factor(mdataset$Block)

head(mdataset)

## Genotype Block Plot Trait Y

## 1 arcobaleno 1 1 TKW 44.5

## 2 arcobaleno 2 2 TKW 42.8

## 3 arcobaleno 3 3 TKW 42.7

## 4 baio 1 4 TKW 40.6

## 5 baio 2 5 TKW 42.7

## 6 baio 3 6 TKW 41.1

tail(mdataset)

## Genotype Block Plot Trait Y

## 157 vesuvio 1 76 Yield 54.37

## 158 vesuvio 2 77 Yield 55.02

## 159 vesuvio 3 78 Yield 53.41

## 160 vitromax 1 79 Yield 54.39

## 161 vitromax 2 80 Yield 50.97

## 162 vitromax 3 81 Yield 48.83

The fixed model is:

Y ~ Trait\*Block

In order to introduce a totally unstructured variance-covariance matrix for the random effect, I used the ‘pdMat’ construct:

random = list(Genotype = pdSymm(~Trait - 1))

Relating to the residuals, heteroscedasticity can be included by using the ‘weight()’ argument and the ‘varIdent’ variance function, which allows a different standard deviation per trait:

weight = varIdent(form = ~1|Trait)

I also allowed the residuals to be correlated within each plot, by using the ‘correlation’ argument and specifying a general ‘corSymm()’ correlation form. Plots are nested within genotypes, thus I used a nesting operator, as follows:

correlation = corSymm(form = ~1|Genotype/Plot)

The final model call is:

mod <- lme(Y ~ Trait\*Block,

random = list(Genotype = pdSymm(~Trait - 1)),

weight = varIdent(form=~1|Trait),

correlation = corCompSymm(form=~1|Genotype/Plot),

data = mdataset)

Retreiving the variance-covariance matrices needs some effort, as the function ‘getVarCov()’ does not appear to work properly with this multistratum model. First of all, we can retreive the variance-covariance matrix for the genotype random effect (G) and the corresponding correlation matrix.

#Variance structure for random effects

obj <- mod

G <- matrix( as.numeric(getVarCov(obj)), 2, 2 )

G

## [,1] [,2]

## [1,] 53.86081 38.83246

## [2,] 38.83246 77.63485

cov2cor(G)

## [,1] [,2]

## [1,] 1.0000000 0.6005237

## [2,] 0.6005237 1.0000000

Second, we can retreive the ‘conditional’ variance-covariance matrix (R), that describes the correlation of errors:

#Conditional variance-covariance matrix (residual error)

V <- corMatrix(obj$modelStruct$corStruct)[[1]] #Correlation for residuals

sds <- 1/varWeights(obj$modelStruct$varStruct)[1:2]

sds <- obj$sigma \* sds #Standard deviations for residuals (one per trait)

R <- t(V \* sds) \* sds #Going from correlation to covariance

R

## [,1] [,2]

## [1,] 4.4718234 0.1635375

## [2,] 0.1635375 6.0939251

cov2cor(R)

## [,1] [,2]

## [1,] 1.00000000 0.03132756

## [2,] 0.03132756 1.00000000

The total correlation matrix is simply obtained as the sum of G and R:

Tr <- G + R

cov2cor(Tr)

## [,1] [,2]

## [1,] 1.0000000 0.5579906

## [2,] 0.5579906 1.0000000

We see that the results are the same as those obtained by using ‘asreml’. Hope these snippets are useful!

**References**

* Butler, D., Cullis, B.R., Gilmour, A., Gogel, B., Thomson, R., 2018. ASReml-r reference manual – version 4. UK.
* Piepho, H.-P., 2018. Allowing for the structure of a designed experiment when estimating and testing trait correlations. The Journal of Agricultural Science 156, 59–70.
* Pinheiro, J.C., Bates, D.M., 2000. Mixed-effects models in s and s-plus. Springer-Verlag Inc., New York.
* Wickham, H., 2007. Reshaping data with the reshape package. Journal of Statistical Software 21.