## File S1: R-Scripts used to fit models

This document illustrates for the wheat data set how the models were fitted in R. To begin, the required R packages are loaded as well as the wheat data from the BGLR package. Further, the trait is chosen and a seed for reproducible random number generation is set (Box S1).

**Box S1: Loading required R packages, data, settings**

```
rm(list = ls())
 library(coda)
 library(MCMCpack)
 library(BGLR)
 library(PSMix)
 library(MTM)

# loading data
data(wheat)

# choosing trait/ environment 1
 trait <- 1

# setting seed for random number generation
 set.seed(12345)

# Setting number of iteration and burn in (Note: for
# illustration only small numbers chosen several thousands needed for inferences)
 nIter <- 1100
 burnIn <- 100
```

As next step the genotypic relationship matrix G is constructed from the wheat marker matrix wheat.X and the cluster analysis is conducted defining the two wheat clusters.

**Box S2: Constructing G matrix; defining wheat clusters**

```
# Constructing the G-matrix
 geno <- scale(wheat.X,center=TRUE,scale=TRUE)
 G <- tcrossprod(geno)/ncol(geno)

# clustering
 X2 <- matrix(nrow = (2*nrow(wheat.X)), ncol = ncol(wheat.X))
 for(i in 1:nrow(wheat.X)){
   X2[((i*2)-1),] <- wheat.X[i,]
   X2[(i*2),] <- wheat.X[i,]
 }
 fm<-PSMix(K=2,X2)
 cluster <- fm$AmId
```

For the models A-GBLUP and W-GBLUP the vector of phenotypic traits **y** is needed. For MG-GBLUP a matrix is constructed for the phenotypic values with dimension *n x K*, where *n* is the number of lines and *K* the number of sub-populations. For the wheat data we have *n* = 599 and *K* = 2. The first column of the matrix is filled with the phenotypic values of sub-population 1 (cluster = 1) while the entries of sub-

population 2 are filled with missing values (NA). The second column contains the phenotypic values of sub-population 2 while the entries of sub-population 1 are filled with NA.

**Box S3: Constructing phenotypic input variables for models**

```
# Vector of phenotypic values
 yVec <-wheat.Y[,trait]

# Constructing matrix of phenotypic values for MG-GBLUP
 yMat<-matrix(nrow=599,ncol=2,NA)
 for(i in 1:599){
    yMat[i,cluster[i]]<-yVec[i]
 }
```

Box S4 illustrates how the overall GBLUP model (A-GBLUP) is fitted using the BGLR function. For fitting GBLUP with cluster-specific residual variances, the eigenvalue decomposition of the G matrix is applied and model 'BRR' within the BGLR function is used. Cluster-specific intercepts are fitted as 'fixed' effects (with uninformative flat priors assigned).

**Box S4: Fitting overall GBLUP model (A-GBLUP)**

```
# construct model matrix for cluster-specific intercepts
Xf <- as.matrix(model.matrix(~factor(cluster))[,-1])

EVD <- eigen(G)
T <- EVD$vectors %*% diag(sqrt(EVD$values))
T <- T[, - which(EVD$values < 1e-6)]

# Specifying model parameters
ETA <- list(list(X=Xf, model='FIXED'), list(X=T, model='BRR', df0=4, S0=3))

# Running the model in BGLR
  fm0 <- BGLR(y=yVec, ETA=ETA, df0=4, S0=3, groups=as.factor(cluster), nIter=nIter,
              burnIn=burnIn, thin=2, saveAt='acrossGroup_GBLUP_')
```

Fitting the stratified within-group GBLUP analysis (W-GBLUP) is illustrated in BoxS5. Here, two separate models are run within BGLR for the two wheat clusters. For fitting GBLUP, the 'RKHS' model with the G matrix as kernel is used within BGLR.

**Box S5: Fitting within-group GBLUP analysis (W-GBLUP)**

```
fm1 <- list()

# Fitting GBLUP model for cluster 1
 set1 <- cluster==1

 ETA <- list(list(K=G[set1,set1], model='RKHS', df0=4, S0=3))

 fm1[[1]] <- BGLR(y=yVec[set1], ETA=ETA, nIter=nIter, burnIn=burnIn,
                  df0=4, S0=3, thin=2, saveAt='withinGroup_GBLUP_1_')

# Fitting GBLUP model for cluster 2
 set2 <- cluster==2

 ETA <- list(list(K=G[set2,set2],model='RKHS', df0=4, S0=3))

 fm1[[2]] <-BGLR(y=yVec[set2], ETA=ETA, nIter=nIter, burnIn=burnIn,
                 df0=4, S0=3, thin=2, saveAt='withinGroup_GBLUP_2_')
```

The multivariate model (MG-GBLUP) is fitted using function MTM in R. Box S6 illustrates the model specification. The R code of the MTM function is provided in File S2.

**Box S6: Fitting the multivariate model (MG-GBLUP)**

```
K <- list(list(K=G, COV=list( type='UN',df0=5,S0=diag(4,ncol(yMat)))))

resCov <-list(type='DIAG', df0=rep(4,ncol(yMat)), S0=rep(3,ncol(yMat)))

fm12 <- MTM(Y=yMat, K=K, resCov=resCov, nIter=nIter,
            burnIn=burnIn, thin=2, saveAt='multi_group')
```

In Box S7 we show how to calculate posterior means and standard deviations from model outputs of A-GBLUP.

**Box S7: Extracting results of A-GBLUP**

```
# read in MCMC samples of genotypic and residual variance
 sampG <- read.table("acrossGroup_GBLUP_ETA_2_varU.dat")
 sampE  <- read.table("acrossGroup_GBLUP_varE.dat")

 burnInT <- burnIn/2 # no of thinned burnIn samples

# defining vectors for variance components and standard errors
 VarE <- vector(length = 2)
 h2 <- vector(length = 2)
 sdVarE <- vector(length = 2)
 sdh2 <- vector(length = 2)

# calculate posterior mean and standard deviation of genotypic
# variance from post-burnIn samples
 VarG <-  mean(sampG[-1:-burnInT,1])
 sdVarG <- sd(sampG[-1:-burnInT,1])

# calculate posterior mean and standard deviation of residual
# variance from post-burnIn samples
 VarE <-  colMeans(sampE[-1:-burnInT,])
 sdVarE <- apply(sampE[-1:-burnInT,], 2, sd)

# calculate posterior mean and standard deviation of
# heritabilities
 h2[1] <- mean(sampG[-1:-burnInT,1]/(sampG[-1:-burnInT,1] + sampE[-1:-burnInT,1]))
 h2[2] <- mean(sampG[-1:-burnInT,1]/(sampG[-1:-burnInT,1] + sampE[-1:-burnInT,2]))
 sdh2[1] <- sd(sampG[-1:-burnInT,1]/(sampG[-1:-burnInT,1] + sampE[-1:-burnInT,1]))
 sdh2[2] <- sd(sampG[-1:-burnInT,1]/(sampG[-1:-burnInT,1] + sampE[-1:-burnInT,2]))
```

In Box S8 we show how to calculate posterior means and standard deviations from model outputs of W-GBLUP.

**Box S8: Extracting results of W-GBLUP**

```
# defining vectors for variance components and standard errors
 VarG <- vector(length = 2)
 VarE <- vector(length = 2)
 h2 <- vector(length = 2)
 sdVarG <- vector(length = 2)
 sdVarE <- vector(length = 2)
 sdh2 <- vector(length = 2)

 burnInT <- burnIn/2 # no of thinned burnIn samples

# loop for calculating var. comp. and s.e. for both clusters
 for(ng in 1:2){
  sampG <- read.table(paste("withinGroup_GBLUP_", ng,"_ETA_1_varU.dat", sep =""))
  sampE  <- read.table(paste("withinGroup_GBLUP_", ng,"_varE.dat", sep =""))
  VarG[ng] <-  mean(sampG[-1:-burnInT,1])
  sdVarG[ng] <- sd(sampG[-1:-burnInT,1])
  VarE[ng] <-  mean(sampE[-1:-burnInT,1])
  sdVarE[ng] <- sd(sampE[-1:-burnInT,1])
  h2[ng] <- mean(sampG[-1:-burnInT,1]/(sampG[-1:-burnInT,1] + sampE[-1:-burnInT,1]))
  sdh2[ng] <- sd(sampG[-1:-burnInT,1]/(sampG[-1:-burnInT,1] + sampE[-1:-burnInT,1]))
 }
```

In Box S9 we show how to calculate posterior means and standard deviations from model outputs of MG-GBLUP.

**Box S9: Extracting results of MG-GBLUP**

```
# read in MCMC samples of residual var-cov matrix R
 sampE  <- read.table("multi_groupR.dat")

# calculate residual variances from post-burn-in samples
 burnInT <- burnIn/2
 VarE <- diag(xpnd(unlist(colMeans(sampE[-1:-burnInT,]))))
# calculate posterior standard deviation of residual variances
 sdVarE <- diag(xpnd(apply(sampE[-1:-burnInT,],2, sd)))

# read in MCMC samples of the genomic var-cov matrix Σ_g
 sampG1 <- read.table("multi_groupG_1.dat")

# constructing lists to collect covariance and correlation
# estimates from every iteration
 CorMatSamples <- list()
 CovMatSamples <- list()
 nIterT <- (nIter-burnIn)/2 # nr of thinned post-burnIn
                            # samples

# constructing matrix to collect heritability estimates from
# every iteration
 h2vec <- matrix(nrow= nIterT, ncol=2)

# extract heritability, genomic covariance, and correlation
# estimates from every iteration
```

```
 for(j in 1:nIterT){
   mat <- xpnd(unlist(sampG1[j+burnInT, ]))
   CovMatSamples[[j]] <- mat
   CorMatSamples[[j]] <- cov2cor(mat)
   h2vec[j,] <- diag(mat)/(diag(mat)+ diag(xpnd(unlist(sampE[j+burnInT,]))))
}

# calculate posterior means of heritabilities
h2 <- colMeans(h2vec)

# calculate posterior standard deviations of heritabilities
 h2SD <- sqrt(colMeans(h2vec^2)-colMeans(h2vec)^2)

# calculate posterior standard deviations of covariance and
# correlation estimates
 CorMatSamplessquared <- lapply(CorMatSamples, function(x){x^2})
 CovMatSamplessquared <- lapply(CovMatSamples, function(x){x^2})
 SDCov <- sqrt(((Reduce('+', CovMatSamplessquared)/nIterT)
         -(Reduce('+',     CovMatSamples)/ nIterT)^2) *(nIterT/(nIterT-1)))
 SDCor <- sqrt(((Reduce('+', CorMatSamplessquared)/nIterT)
         -(Reduce('+',    CorMatSamples)/nIterT)^2) *(nIterT/(nIterT-1)))
```

Box S10 illustrates how to extract the estimated genotypic effects from the model outputs

**Box S10: Extract estimates from model outputs**

```
# Extracting predicted genomic effects of Model 1
 uHatMod1 <-  T %*% fm0$ETA[[2]]$b

# Extracting predicted genomic effects of Model 2
 uHatMod2_cluster1 <- fm1[[1]]$ETA[[1]]$u
 uHatMod2_cluster2 <- fm1[[2]]$ETA[[1]]$u

# Extracting predicted genomic effects of Model 3
 uHatMod3  <-  fm12$K[[1]]$U
 uHatMod3_cluster1 <- uHatMod3[cluster == 1, 1]
 uHatMod3_cluster2 <- uHatMod3[cluster == 2, 2]
```

In the following BoxS11 we illustrate the performance of the cross-validation for the wheat data. Here, the vector and matrix of phenotypic values generated in Box S6 is used, as well as the genotypic relationship matrix generated in Box S2.

**Box S11: Illustration of cross-validation performance**

```
 rep <- 1; tstProp <- 0.5 # choose number of replicates and size of the testing set
# set specific seed for current replication
 seeds <- sample( (1:10000+10000),size=500,replace=FALSE); set.seed(seeds[rep])

# setting phenotypes of test set to NA
 tst1 <- sample(which(!is.na(yMat[,1])),
                size=(tstProp*length(which(!is.na(yMat[,1])))),replace=FALSE)
 tst2 <- sample(which(!is.na(yMat[,2])),
                size=(tstProp*length(which(!is.na(yMat[,2])))),replace=FALSE)
 yMat[tst1,1] <- NA;  yMat[tst2,2] <- NA; yVec[c(tst1,tst2)]<-NA
```

```
# fitting models
  # Across-groups G-BLUP (A-GBLUP)

  Xf <- as.matrix(model.matrix(~factor(cluster))[,-1])

  fm0 <- BGLR(y=yVec,ETA=list(list(X= Xf, model = 'FIXED'),
              list(X=T,model='BRR', df0=4, S0=3)),
              df0=4, S0=3, groups=as.factor(cluster), nIter=nIter, burnIn=burnIn,
              saveAt='acrossGroup_GBLUP_')


 # within-group GBLUP (W-GBLUP)
 set1 <- cluster==1
 fm1 <- BGLR(y=yVec[set1], ETA=list(list(K=G[set1,set1],
              model='RKHS', df0=4, S0=3)), nIter=nIter, burnIn=burnIn,
              df0=4, S0=3, thin=2, saveAt='withinGroup_GBLUP_1_')

 set2<-cluster==2
 fm2 <- BGLR(y=yVec[set2],ETA=list(list(K=G[set2,set2],model='RKHS', df0=4, S0=3)),
              nIter=nIter, burnIn=burnIn, df0=4, S0=3, thin=2,
              saveAt='withinGroup_GBLUP_2_')

 # Multi-group GBLUP (MG-GBLUP)
  fm12<-MTM(Y=yMat, K=list(list(K=G, COV=
list(type='UN',df0=5,S0=diag(4,ncol(yMat))))),
             resCov=list(type='DIAG',df0=rep(4,ncol(yMat)), S0=rep(3,ncol(yMat))),
             nIter=nIter, burnIn=burnIn, saveAt='multi_group')

# Calculate predictive abilities per cluster and model
 y <- wheat.Y[,trait]
# predictive correlation Model 1, cluster 1
 cor(fm0$yHat[tst1],y[tst1])
# predictive correlation Model 1, cluster 2
 cor(fm0$yHat[tst2],y[tst2])

# predictive correlation Model 2, cluster 1
 cor(fm1$yHat[fm1$whichNa],y[set1][fm1$whichNa])
# predictive correlation Model 2, cluster 2
 cor(fm2$yHat[fm2$whichNa],y[set2][fm2$whichNa])
# predictive correlation Model 3, cluster 1
 cor(fm12$YHat[tst1,1],y[tst1])
# predictive correlation Model 3, cluster 2
 cor(fm12$YHat[tst2,2],y[tst2])
```