Genomic prediction with rrBLUP 4

Jeffrey Endelman

June 15, 2013

This document shows how to use several new features that have been added to the rrBLUP package since the original publication (Endelman 2011).

The basic core of the package is still `mixed.solve`, which solves mixed models with one variance component other than the residual error. The function estimates the two variance components by either ML or REML using my own implementation of the spectral decomposition algorithm described by Kang et al. (2008). The inverse phenotypic covariance matrix is readily constructed during this process, which is then used to calculate the BLUE and BLUP solutions for the fixed and random effects, respectively (Searle et al. 1992). In Endelman (2011) I showed how `mixed.solve` can be used for genomic prediction based either on modeling the markers as random effects or modeling the lines as random effects.

Version 4 of the package has been designed around the functions `A.mat` and `kin.blup`.

A.mat

The function `A.mat` estimates the realized relationship matrix (A) from markers. For high-density markers without missing data, `A.mat` uses the first formula suggested by VanRaden (2008):

$$\mathbf{A} = \frac{\mathbf{WW}'}{2\sum_k p_k q_k} \qquad [1]$$

where the allele calls in **W** are centered by the population mean. Endelman and Jannink (2012) have shown that this formula has the desirable property that its mean diagonal element is 1+*f*, where *f* is the inbreeding coefficient for the current population.

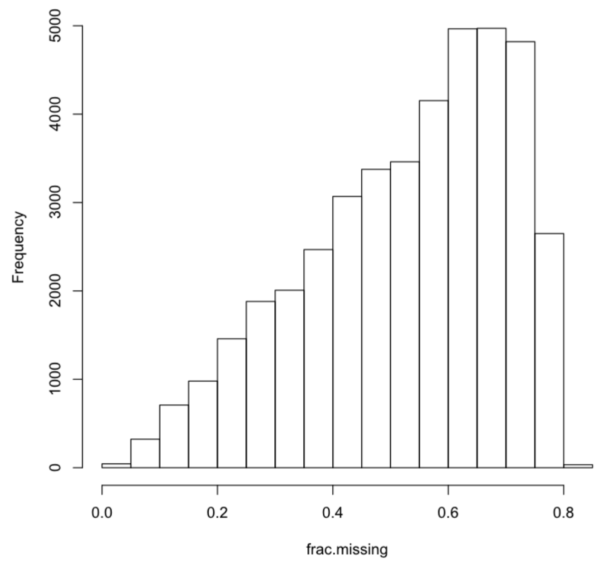
Missing marker data

When genotypes are missing, `A.mat` has two options for imputation. One is to replace missing data with the population mean for that marker, which is adequate for the low levels of missing data typically encountered with SNP chips. For genotyping-by-sequencing (GBS) markers, the levels of missing data can be much higher. For this situation, `A.mat` can estimate the relationship matrix using an EM algorithm based on the multivariate normal distribution (Poland et al. 2012).

To illustrate the EM algorithm, I downloaded the GBS data in the supplemental file published by Poland et al. (2012) from
https://www.crops.org/publications/tpg/supplements/5/tpg12-06-0006-dataset-s2.gz

The following R code reads the file and converts the marker calls to the {-1,0,1} format expected by rrBLUP:

```
> GBS<-read.csv("tpg12-06-0006-dataset-s2",header=T,as.is=TRUE,row.names=1)
> parse.GBS <- function(x) {
+ unique.x <- unique(x)
+ alleles <- setdiff(unique.x,union("H","N"))
+ y <- rep(0,length(x))
+ y[which(x==alleles[1])] <- -1
+ y[which(x==alleles[2])] <- 1
+ y[which(x=="N")] <- NA
+ return(y)
+ }
> X <- apply(GBS[,-c(1:3)],1,parse.GBS)
> dim(X)  #lines by markers
[1]   254 41371
> frac.missing <- apply(X,2,function(z){length(which(is.na(z)))/length(z)})
> length(which(frac.missing<0.5))
[1] 16030
> hist(frac.missing)
```



There are 16K markers with less than 50% missing data, which is more than enough to estimate the relationship matrix for 254 lines. Because I am running this code on a UNIX-compliant system with multiple processors, I can use 12 cores to speed up the EM algorithm:

```
> library(rrBLUP)
> system.time(A1 <- A.mat(X,impute.method="EM",n.core=12,max.missing=0.5))
[1] "A.mat converging:"
[1] 0.132
[1] 0.0458
[1] 0.0238
[1] 0.0151
   user  system elapsed
529.710  97.352  58.794
```

The EM algorithm displays the convergence sequence as it proceeds, which represents the root-mean-squared error of the relationship coefficients. The default stopping criterion is 0.02, but this can be changed with the parameter “tol” (type ?A.mat for more information). In this case the algorithm stopped when it reached 0.0151, which required just under one minute. If only 1 core is available, simply omit the n.core option and by default only 1 core will be used.

To impute with the mean instead of the EM algorithm, the syntax is

```
> system.time(A2 <- A.mat(X,max.missing=0.5))
```

```
   user  system elapsed
  3.812   0.393   4.203
```

Imputing with the mean is definitely faster, and in many cases it seems to perform as well as the EM algorithm and other more sophisticated methods with respect to GEBV accuracy. However, the breeding values tend to be more biased when imputing with the mean compared to the EM algorithm (Poland et al. 2012). A comparison of the mean diagonal elements of the two matrices shows that with the EM algorithm, the result is closer to the expectation $1+f \approx 2$ given the roughly 1% heterozygosity in the marker calls:

```
> round(mean(diag(A2)),2)  #imputed with mean
[1] 1.27
> round(mean(diag(A1)),2)  #imputed with EM
[1] 1.82
```

Shrinkage estimation of A

Another unique feature of `A.mat` is shrinkage estimation, which can be useful with low-density markers, e.g., from a 384 SNP chip. When the number of lines is comparable to or more than the number of markers, Equation 1 may not be an optimal estimator of the relationship matrix. Endelman and Jannink (2012) proposed shrinking the estimate toward $(1+f)\mathbf{I}$, with the shrinkage intensity chosen to minimize the mean-squared error.

To illustrate, I will use a set of 599 wheat lines genotyped at 1279 DArT markers from the BLR package (Pérez et al. 2010):

```
> library(BLR)
Loading required package: SuppDists
Package 'BLR', 1.3 (2012-03-18).
Type 'help(BLR)' for summary information
> data(wheat)
> M <- 2*X-1  #convert markers to {-1,1}
> dim(M)
[1]  599 1279
> A1 <- A.mat(M,shrink=TRUE)
[1] "Shrinkage intensity: 0.03"
> A2 <- A.mat(M[,sample(1:1279,384)],shrink=TRUE)
[1] "Shrinkage intensity: 0.1"
```

```
> A3 <- A.mat(M[,sample(1:1279,192)],shrink=TRUE)
[1] "Shrinkage intensity: 0.17"
```

As shown in the above example, the shrinkage intensity, which ranges from 0 (no shrinkage) to 1 (complete shrinkage), increases as marker density decreases. With all 1279 markers, very little shrinkage (3%) was used, while with a random set of 384 or 192 markers the shrinkage intensity increased to 10% and 17%, respectively.

`kin.blup`

In Endelman (2011) I introduced a wrapper for `mixed.solve` called `kinship.BLUP` that was designed for line-based prediction, using either the additive genetic model or a Gaussian kernel. However, this wrapper was not as convenient as it could have been, so I have designed a new function to replace it called `kin.blup`. Instead of requiring the user to make the design matrices, this new function does that part automatically from a data frame. Another difference is that with `kin.blup` the user passes the relationship matrix rather than the markers, which allows greater control over how the **A** matrix is calculated (see section on `A.mat`).

To illustrate the basic functionality, I will continue with the wheat example from the previous section, in which the relationship matrix `A1` was estimated with all markers. The 599 lines have been divided into 10 sets for cross-validation in the BLR package. To predict breeding values for the lines in set 1 using phenotypes for the lines in sets 2–10, first the phenotypes and corresponding genotypes identifiers must be assembled into a data frame:

```
> test <- which(sets==1)
> yNA <- Y[,1]  #grain yield in environment 1
> yNA[test] <- NA  #mask yields for validation set
> data1 <- data.frame(y=yNA,gid=1:599)
```

To mask the phenotypes for the validation set, it is sufficient to set their values equal to NA, as in the above example. A minimalist data set will have two columns, one with the phenotypes, and one with the genotype labels. After making sure the genotype labels in the data frame correspond to the rownames of the relationship matrix, we can make the genomic predictions:

```
> rownames(A1) <- 1:599
> ans1 <- kin.blup(data1,K=A1,geno="gid",pheno="y")
> str(ans1)
List of 4
 $ Vg   : num 0.32
 $ Ve   : num 0.539
 $ g    : num [1:599(1d)] 0.531 -0.422 -0.321 0.54 0.615 ...
  ..- attr(*, "dimnames")=List of 1
  .. ..$ : chr [1:599] "1" "2" "3" "4" ...
 $ resid: num [1:542, 1] 1.142 0.171 0.665 0.248 0.385 ...
```

As shown in the above example, the relationship matrix has been passed as parameter “K” (for kinship) to `kin.blup`, along with the names of the phenotype and genotype labels in the data frame. The function returns the REML estimates of the variance components ($Vg, $Ve) and the BLUP solution for the genetic values ($g), which in this case are breeding values. The residuals from the mixed model are also returned. As shown above, BLUP values were returned for all 599 entries in the relationship matrix, even though 10% of these lines had no phenotype (due to masking). The order of the BLUPs follows the order in the K matrix (the array is also named).

To estimate the GEBV accuracy, the predictions for the validation set can be correlated against the masked phenotypes:

```
> round(cor(ans1$g[test],Y[test,1]),2)
[1] 0.49
```

For predictions with the Gaussian kernel, instead of passing the relationship matrix for K, use the Euclidean distance and set the GAUSS flag to TRUE:

```
> D <- as.matrix(dist(M))  #Euclidean distance
> system.time(ans2 <- kin.blup(data1,K=D,GAUSS=TRUE,geno="gid",pheno="y"))
user  system elapsed
 20.693   0.338  21.028
> system.time(ans2<-kin.blup(data1,K=D,GAUSS=TRUE,geno="gid",pheno="y",n.core=10))
user  system elapsed
 45.767   7.685   3.974
> round(cor(ans2$g[test],Y[test,1]),2)
[1] 0.63
```

As shown in this example, using multiple cores can speed up the Gaussian kernel prediction. The prediction accuracy with the Gaussian kernel was 0.63, which is higher than with the additive relationship matrix. This result is not identical to that in Endelman (2011) because the grid points used to determine the optimal scale parameter are not identical; type ?kin.blup for more information about how to set the grid points.

Multi-environment trials

The `kin.blup` function has the ability to handle repeated measurements taken over different environments. The wheat data set from the BLR package has average yield measurements for the 599 lines in 4 environments. Environments 2–4 are correlated and for this example will be considered as representing a target population of environments for a breeding program. The following code creates an unbalanced data set with partial replication across environments:

```
> y <- c(Y[1:400,2],Y[101:400,3],Y[201:500,4])
> env <- c(rep(2,400),rep(3,300),rep(4,300))
> gid <- c(1:400,101:400,201:500)
> data2 <- data.frame(y=y,env=env,gid=gid)
> nrow(data2)
[1] 1000
```

To model the main effect of the environment as a fixed effect in the prediction model, the name of the data frame column is passed to the function:

```
> system.time(ans <- kin.blup(data2,K=A1,geno="gid",pheno="y",fixed="env"))
   user  system elapsed
  8.566   0.081   8.648
> round(cor(ans$g[501:599],rowMeans(Y[501:599,2:4])),3)  #accuracy
[1] 0.524
```

When there are multiple fixed effects to model, such as year and location, simply pass an array of the relevant column names, e.g., `fixed=c("year","location")`.

The above example is a one-step prediction, which is more computationally demanding than a two-step approach in which line means are calculated first. For unbalanced data, `kin.blup` can make predictions with the speed of a two-stage approach while retaining information about the different levels of replication. This is possible with the `reduce=TRUE` option, which transforms the mixed model to have dimension equal to the number of lines (see reference manual for details):

```
> system.time(ans2<-kin.blup(data2,K=A1,geno="gid",pheno="y",fixed="env",reduce=TRUE))
   user  system elapsed 
  1.795   0.019   1.814 
> round(cor(ans2$g[501:599],ans$g[501:599]),3)
[1] 0.965
> round(cor(ans2$g[501:599],rowMeans(Y[501:599,2:4])),3)  #accuracy
[1] 0.513
```

In the above example, the computing time went down nearly fivefold with the reduction method. The predictions with the two methods were very similar ($r = 0.96$), but there was slightly higher accuracy without reduction in this example.

References

Endelman, J.B. 2011. Ridge regression and other kernels for genomic selection with R package rrBLUP. Plant Genome 4:250–255. doi:10.3835/plantgenome2011.08.0024

Endelman, J.B., and J.-L. Jannink. 2012. Shrinkage estimation of the realized relationship matrix. G3:Genes, Genomes, Genetics. 2:1405-1413. doi:10.1534/g3.112.004259

Kang et al. 2008. Efficient control of population structure in model organism association mapping. Genetics 178:1709–1723.

Pérez et al. 2010. Genomic-enabled prediction based on molecular markers and pedigree using the Bayesian Linear Regression package in R. Plant Genome 3:106–116.

Poland, J., J. Endelman et al. 2012. Genomic selection in wheat breeding using genotyping-by-sequencing. Plant Genome 5:103–113. doi: 10.3835/plantgenome2012.06.0006

Searle et al. 1992. Variance Components. John Wiley & Sons, Hoboken.

VanRaden, P.M. 2008. Efficient methods to compute genomic predictions. J. Dairy Science 91:4414–4423.