Module 3: Genomic Prediction Models: Basic to Advanced

Fundamentals of Genomic Prediction and Data-Drive Crop Breeding

(August 4-8, 2025)

Waseem Hussain
Senior Scientist-I
International Rice Research Institute
Rice Breeding Innovations Platfrom
waseem.hussain@irri.org
whussain2.github.io

Mahender Anumalla
Scientist-I
International Rice Research Institute
South-Asian Hub, Hyderabad
m.anumalla@irri.org

July 22, 2025

Contents

Load the Libraries

```
> # Load the Required Libraries
>   rm(list=ls()) # Remove previous work
>   library(rrBLUP)
>   library(BGLR)
>   library(AGHmatrix)
>   library(ggplot2)
>   library(DT)
>   library(cvTools)
```

In this section, We will use two R packages **rrBLUP** and BGLR for performing genomic predictions. For more details please click on the links **rrBLUP** and **BGLR**.

Extensive details on BGLR package with examples can be found here **Click Me**

Data Sets: Phenotypes and Genotypes

Phenotype Data

The data set we will use is from IRRI's drought breeding program that has 616 genotypes. The phenotypic data is analyzed, and we have the adjusted means as **BLUEs** for four traits: Yield under non-stress (blues_ns); Yield under stress (blues_s), drought indices (MPI and STI). The data set is in folder in the ***.rds** format.

```
> # Read the phenotypic data set
>   pheno<-readRDS("./Data/pheno.filtered.rds")
>   pheno<-pheno[,1:5] # Removing missing columns
```

```
> # View data as table
>   head(pheno)
```

Designation	blues_s	blues_ns	MPI	YSI
IR129391.B.12.B.3.1	493.2188	7271.837	3882.528	0.0678259
IR129391.B.19.B.2.1	227.0740	7714.415	3970.745	0.0294350
IR129391.B.44.B.4.1	364.1401	6474.481	3419.311	0.0562424
IR129391.B.46.B.1.1	1460.8218	5650.479	3555.651	0.2585306
IR129391.B.6.B.4.1	2552.4985	6162.866	4357.682	0.4141739
IR129391.B.9.B.4.1	2284.5110	6790.909	4537.710	0.3364073

Genotypic Data

The marker data is **1KRiCA** with 887 high quality SNP markers. The data is filtered and low quality markers and minor allele frequency markers has been already removed. The data is on **1**, **0** and **-1** format. For example for SNP1 if we have allele call as **A/T**, then we will have 3 types of genotypes **AA**, **TT** and **AT**. For this SNP, **[1==AA, 0==AT and TT==1**.

More on filtering on How to filter low quality markers can be found here **How to Filter the SNP markers**

```
> # Genotype data
>   geno<-readRDS("./Data/geno.oyt.filtered.rds")
>   dim(geno) # Check dimensions
```

```
[1] 616 887
```

```
>   kable(head(geno)[1:6,1:2])
```

	IRRI_SNP0001_CHR01_194844	IRRI_SNP0002_CHR01_375814
IR129391.B.12.B.3.1	-1	-1
IR129391.B.19.B.2.1	-1	-1
IR129391.B.44.B.4.1	1	1
IR129391.B.46.B.1.1	1	1
IR129391.B.6.B.4.1	-1	-1
IR129391.B.9.B.4.1	-1	-1

```
>   geno[is.na(geno)] <-0
```

RRBLUP Model

The baseline Random regression BLUP (RRBLUP) mixed model is given as follows:

$$y = 1u + Z\beta + e$$

where y is the vector of BLUEs or response variable; u represents the mean (fixed); β is the vector of random marker effects, where $\beta \sim N(0, I\sigma^2_\beta)$ and σ^2_β is the marker variance; and e is residual, where $e \sim N(0, I\sigma^2_\epsilon)$, is the residual variance. Z is the design matrix of m markers.

In the rrBLUP package, we will use the matrix of markers directly. We will use function ***mixed.solve*** to run the RRBLUP model in **rrBLUP** package and also run it in **BGLR** R package.

After running the model in rrBLUP it will retrun the list of 5 items: **1**) Vu variance of markers, **2**) Ve error variance, **3**) $\beta(BLUES)$, **4**) μ (BLUPs) or marker effects and **5**) LLmaximized log-likelihood.

In BGLR R Package). In this package, we need an argument "ETA". In the ETA, we set relationship matrix (Z) or markers data (X), and also specify the models such as *BRR*, *BayesA*, *BayesB*, *BayesC*, etc.

Run the Model in rrBLUP Package

```
> # Mixed model in rrBLUP
>   model.rrblup<- mixed.solve(y =pheno[,2], Z = geno) # RR-BLUP
```

Get Marker Efects and Estimated Breeding Values (EBVs)

```
> # Get Marker effects
>   kable((model.rrblup$u)[1:5])
```

	x
IRRI_SNP0001_CHR01_194844	-3.8922431
IRRI_SNP0002_CHR01_375814	-3.0734353
IRRI_SNP0003_CHR01_717702	-6.1305573
IRRI_SNP0004_CHR01_932866	-5.9207302
IRRI_SNP0005_CHR01_1044946	-0.2910026

```
> # Get GEBVs (Multiple marker effects by Matrix)
>   GEBVs = data.frame(EBVs=geno %*% model.rrblup$u)
>   kable(head(GEBVs))
```

	EBVs
IR129391.B.12.B.3.1	-170.7255
IR129391.B.19.B.2.1	-148.3014
IR129391.B.44.B.4.1	-213.8200
IR129391.B.46.B.1.1	333.3299
IR129391.B.6.B.4.1	152.2292
IR129391.B.9.B.4.1	427.4063

Run Model in BGLR Package

```
> # Set the ETA
> ###----------- Linear predictor
> ETA = list(A = list(X = geno,    # X (Markers) in ETA
+                     model = 'BRR'))  # Specify the model
>
> # Phenotypic data
> Y = pheno[,2]
>
> # Run Model for trait 1
> model.rrblupB = BGLR(y=Y,          # Phenotypic data
+                 ETA=ETA,          # Model ETA
```

```
+                 nIter=1200,        # Number of iterations for the model
+                 burnIn=120,        # Burnin iterations
+                 thin=5,            # Sampling throughout iterations
+                 verbose = FALSE)
```

Get Marker Effects and EBVs

We will extract marker effects (u; random), variance components and Estimated Breeding Values

```
> # Mean estimated
>   model.rrblupB$mu
```

```
[1] 1277.123
```

```
> #Variance component for SNP  marker effects
>   model.rrblupB$ETA[[1]]$varB
```

```
[1] 222.2338
```

```
> #Variance component for the residual effect
>   model.rrblupB$varE
```

```
[1] 271100.7
```

```
> # SNP effects
>   length(model.rrblupB$ETA[[1]]$b) # markers effects
```

```
[1] 887
```

```
> #Get GEBVs
>   GEBVs.B =data.frame(EBVs=geno%*% model.rrblupB$ETA[[1]]$b)
>    kable(head(GEBVs.B))
```

	EBVs
IR129391.B.12.B.3.1	-158.1682
IR129391.B.19.B.2.1	-159.7097
IR129391.B.44.B.4.1	-229.8898
IR129391.B.46.B.1.1	390.5114
IR129391.B.6.B.4.1	233.3308
IR129391.B.9.B.4.1	485.6094

GBLUP Model

- The baseline **GBLUP** model is given as:

$$y = X\beta + Zu + e$$

where y is the vector of BLUEs or response variable; β represents the fixed effects; X is the design matrix of fixed effects, u is the vector of random marker effects, where $u \sim N(0, I\sigma_u^2)$ and σ_u^2 is the marker variance; and e is residuals, where $e \sim N(0, I\sigma_e^2)$. Z is the design matrix of m markers.

In GBLUP we will use **G Matrix** rather than Marker design matrix to fit the model.In GBLUP model u represent the *Genomic Estimated Breeding Values **not marker effects. It is called** BLUP of Breeding Value**. Further in GBLUP the co-variances between breeding values is based on the genomic relationship matrix G which is computed from SNP markers. If we are using pedigree data and constructing pedigree matrix called as numerator relationship matrix A we called Pedigree-BLUP (PBLUP).

Build the G matrix

We will use the **AGHmatrix** R package to build G matrix (Recall the session on Relationship matrices). We need to do conversion and change the format from **1,0,-1** to **2,1,0**.

```
> # Marker data as matrix
>   geno1<-as.matrix(geno)
>   geno1[geno1 ==1] <- 2 # Conversion
>   geno1[geno1 ==-1] <- 1 # Conversion
>   geno1[geno1 ==0] <- NA # Conversion
> # Build the G matrix
>   G_matrix = Gmatrix(SNPmatrix=geno1, method = "VanRaden",
+                      ploidy = 2,integer = FALSE, missingValue =NA)
```

```
Initial data:
    Number of Individuals: 616
    Number of Markers: 887

Missing data check:
    Total SNPs: 887
     2 SNPs dropped due to missing data threshold of 0.5
    Total of: 885  SNPs

MAF check:
    No SNPs with MAF below 0

Heterozigosity data check:
    No SNPs with heterozygosity, missing threshold of =  0

Summary check:
    Initial:  887 SNPs
    Final:  885  SNPs ( 2  SNPs removed)

Completed! Time = 0.19  seconds
```

Run the Model in rrBLUP Package

```
> # Run the GBLUP model
>   model.gblup<- mixed.solve(y = pheno[,2], Z = G_matrix) # G-BLUP
>   gebvs.gblup<-data.frame(Gebvs=model.gblup$u)
>   # Add intercept
>   intercept<-model.gblup$beta
>   gebvs.gblup$Gebvs<-gebvs.gblup$Gebvs+intercept
>   kable(head(gebvs.gblup))
```

	Gebvs
IR129391.B.12.B.3.1	1155.457
IR129391.B.19.B.2.1	1158.937
IR129391.B.44.B.4.1	1164.518
IR129391.B.46.B.1.1	1216.786
IR129391.B.6.B.4.1	1201.791
IR129391.B.9.B.4.1	1222.696

Bayesian Models

BayesA

In BayesA, prior density of the marker effects follows scaled-t distribution $\beta_j(0, \sigma^2_{\beta_j})$

```
> # Pheno
>   Y = as.matrix(pheno[,2])
> # Model
>   model.BayesA<- BGLR(y=Y,# Phenotypic data blues non-stress
+               ETA=list(list(X=geno, model='BayesA')),          # Model Bayes A
+               nIter=1200,       # Number of iterations for the model
+               burnIn=120,       # Burnin iterations
+               thin=5,
+               verbose = FALSE)           # Sampling throughout iterations
> # Mean estimated
>   model.BayesA$mu
```

```
[1] 1269.886
```

```
> # Variance component for the residual effect
>   model.BayesA$varE
```

```
[1] 286393.8
```

```
> # SNP effects
>   length(model.BayesA$ETA[[1]]$b) #'b' slot
```

```
[1] 887
```

```
> # Assignment: How to get Estimated Breeding Values
```

BayesB

Fraction of markers with zero effect. The distribution is given as $\beta_j(0, \sigma^2_{\beta_j})$, where $\beta_j = 0$ with probability π and $1 - \pi$

```
> # Pheno
>   Y = as.matrix(pheno[,2])
> #Model
>   model.BayesB<- BGLR(y=Y,# Phenotypic data blues non-stress
+               ETA=list(list(X=geno, model='BayesB')),          # Model Bayes A
+               nIter=1200,       # Number of iterations for the model
+               burnIn=120,       # Burnin iterations
+               thin=5,
+               verbose = FALSE)           # Sampling throughout iterations
>
> # Mean estimated
>   model.BayesB$mu
```

```
[1] 1287.94
```

```
> # Variance component for the residual effect
>   model.BayesB$varE
```

```
[1] 280748.1
```

```
> # SNP effects
>   length(model.BayesB$ETA[[1]]$b) #'b' slot
```

```
[1] 887
```

BayesC

The prior densities of the marker effects for BayesC model are assumed to be Gaussian mixture: $\beta_j(0, \sigma^2_{\beta_j})$, where $\beta_j = 0$ with probability π and $\beta_j = 0$, with probability $1 - \pi$, with π assumed to follow a BETA distribution.

```
> # Pheno
>   Y = as.matrix(pheno[,2])
> #Model
>   model.BayesC<- BGLR(y=Y,# Phenotypic data blues non-stress
+               ETA=list(list(X=geno, model='BayesC')),          # Model Bayes A
+               nIter=1200,       # Number of iterations for the model
+               burnIn=120,       # Burnin iterations
+               thin=5,
+               verbose = FALSE)           # Sampling throughout iterations
>
> # Mean estimated
>   model.BayesC$mu
```

```
[1] 1327.464
```

```
> # Variance component for the residual effect
>   model.BayesC$varE
```

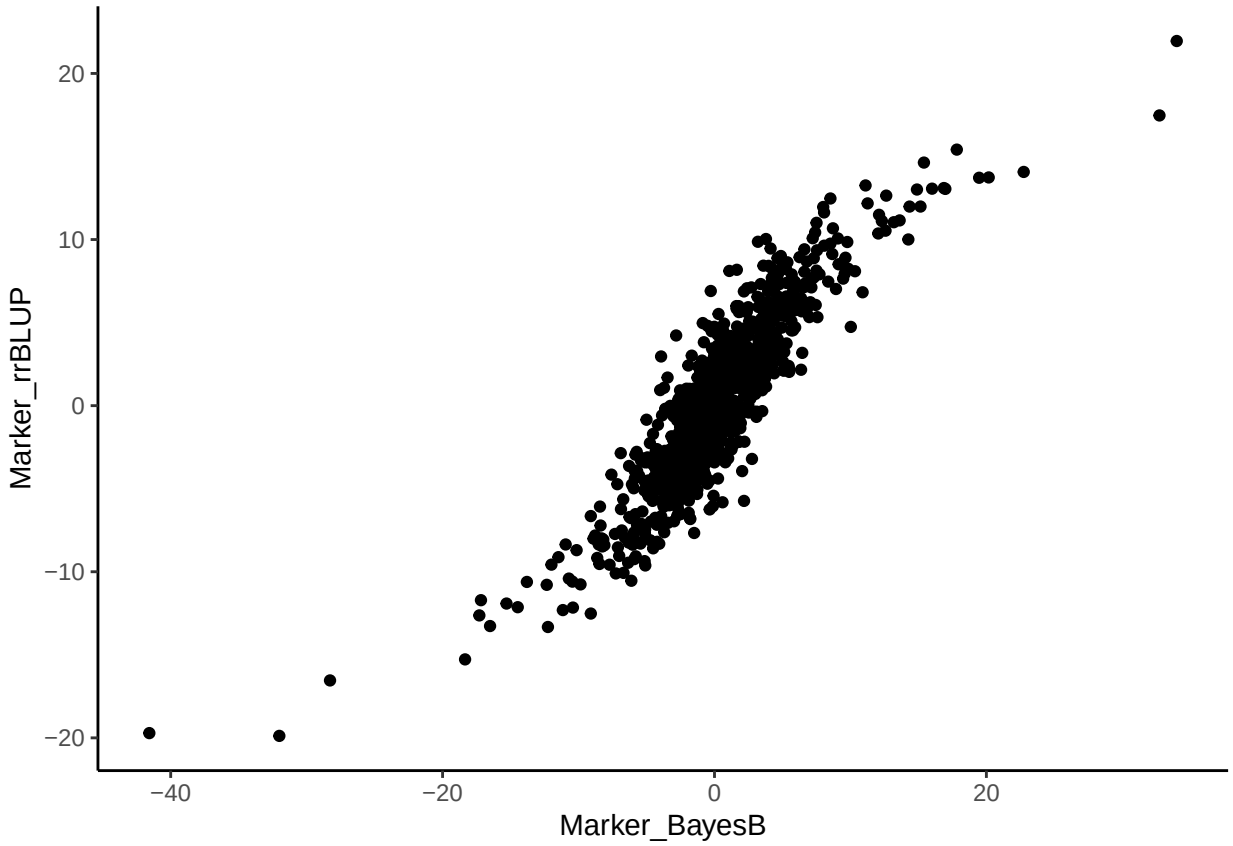
```
[1] 282796.3
```

```
> # SNP effects
>   length(model.BayesC$ETA[[1]]$b) #'b' slot
```

```
[1] 887
```

Comparing the Models RRBLUP vs. BayesB

```
> # Plotting the effects
>   BayesB_rrBLUP = data.frame("Marker_BayesB" = model.BayesB$ETA[[1]]$b,
+                              "Marker_rrBLUP" = model.rrblupB$ETA[[1]]$b)
>
>   ggplot(BayesB_rrBLUP, aes(x = Marker_BayesB, y = Marker_rrBLUP))+
+   geom_point()+
+   theme_classic()
```



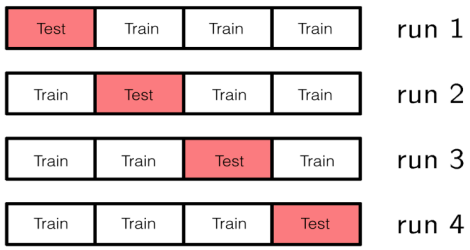
Cross-validation

Cross-validation

(evaluate prediction performance)

- ❖Take model uncertainty into account
- ❖Divide data into training and testing sets
- ❖Train the model in the training set
- ❖Evaluate predictive performance in the testing set
- ❖Predictive correlation: $r = cor(y, y_{predicted})$

K-fold Cross-validation



Cross validation is important is prediction and take model uncertainty into account. Here what we will divide data into training and testing sets. We will use k fold cross validation approach and check *correlation accuracy* $r = cor(y, y_{predicted}$. Recall correlation accuracy is correlation between Ture Values (BLUEs here) and Predicted Values.

What We Will DO

- We will divide data into 4 parts (folds) with 154 observations each. We will check the order of pheno and geno files.
- We will assign each observation to folds. We will repeat it two times. R package cvTools will be used for this. We will add ID column indicating observations.
- We will do CV. We will create **fold.list** for fold assignment, **results** and **Out** to store output from each interation process. All will be done in a loop.
- The output of the code is in **results**, which has Yhat as *GEBVs*. The *out* has summary of prediction accuracy for two replications. Log" is a diagnostic indicating if the order of the predicted breeding values matches the order of the adjusted means, "Ac" is the prediction accuracy (correlation between the GEBV and adjusted means), and "MSPE" is the mean square prediction error (the lower, the better).

```
> # Get the number of folds and replications
>   nfolds = 4 # Four folds
>   nrep = 2
> #  Order Pheno and geno files
>   pheno = pheno[order(pheno$Designation, decreasing = FALSE),]
>   geno= geno[order(row.names(geno)),]
>   all(rownames(geno) == pheno$Designation)
```

```
[1] TRUE
```

```
>   pheno$Designation<-as.factor(pheno$Designation)
> # Create a factor OF idS
>   pheno$ID = factor(1:nrow(pheno))
>   set.seed(100)
> # Create fold assignment using forloop
>   sort<- list()
>   for(a in 1:nrep){
+     for(j in 1:nfolds){
+       folds <- cvFolds(nlevels(pheno$Designation),type = "random", K = 4, R = 1)
+       Sample <- cbind(folds$which,folds$subsets)
+       cv <- split(Sample[,2], f=Sample[,1])
+     }
+     sort[[a]] <- cv
+   }
>   rm(a, folds, j, cv, Sample)
>
> # Save outputs
>   fold.list = sort
>   results = list()
>   Out = list()
> # Get the model and run it
>   ETA = list(list(X = geno, model = "BayesB"))
>   for (z in 1:length(fold.list)) {
+     for (i in 1:nfolds){
+
+ # Training set
+       train_data <- pheno
+ # Validation set
+       train_data[train_data$ID %in% fold.list[[z]][[i]], "blues_ns"] <- NA
+ # Fitting model
+       BB_M <- BGLR(y = train_data$blues_ns, ETA = ETA, nIter = 10000,
```

```
+                     burnIn = 5000, thin = 5, verbose = F)
+ # GEBV
+       Pred <- data.frame(Yhat = BB_M$yHat, G = pheno$ID)
+       rownames(Pred) <- rownames(geno)
+ # Predicted GEBV
+       results[[i]] <- Pred[Pred[, "G"] %in% fold.list[[z]][[i]], ]
+ # Remove
+       rm(BB_M, Pred, train_data)
+     }
+     GEBV <- do.call(rbind, results)
+     GEBV <- GEBV[order(GEBV$G), ]
+ # Log
+     log <- all(GEBV$G == pheno$ID)
+ # GET Results
+     Out[[z]] <- data.frame(
+       Rep = z,
+       Log = log,
+       Ac = round(cor(GEBV$Yhat, pheno$blues_ns, use = "na.or.complete"), 3),
+       MSPE = round(mean(((GEBV$Yhat - pheno$blues_ns)^2), na.rm = T), 3)
+     )
+   }
```

Note: Code adopted and modified from this Source on GitHub

Additional Resources

- Genomic selection in plant breeding: Key factors shaping two decades of progress
- Breeding schemes for the implementation of genomic selection in wheat (Triticum spp .)
- Prospects for Genomewide Selection for Quantitative Traits in Maize
- Genomic Selection in Plant Breeding: Methods, Models, and Perspectives
- Hybrid breeding of rice via genomic selection
- Whole-Genome Regression and Prediction Methods Applied to Plant and Animal Breeding
- Semi-parametric genomic-enabled prediction of genetic values using reproducing kernel Hilbert spaces methods
- Genomic-Assisted Prediction of Genetic Value With Semiparametric Procedures
- Optimal Designs for Genomic Selection in Hybrid Crops
- Genomic Selection for Crop Improvement
- Plant Breeding with Genomic Selection: Gain per Unit Time and Cost
- Genomic selection in plant breeding: from theory to practice
- Genomic prediction in hybrid breeding: I. Optimizing the training set design
- Multivariate Statistical Machine Learning Methods for Genomic Prediction
- Priors in Whole-Genome Regression: The Bayesian Alphabet Returns
- Kernel-based whole-genome prediction of complex traits: a review

For any suggestions or comments, please feel to reach at waseem.hussain@irri.org; and m.anumalla@irri.org