Module 3: End-to-End Genomic Selection: Workflow in R

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

August 1, 2025

Contents

Background Information

Here in this module we will go for an end-to-end process how to perform a genomic selection in applied breeding Program.

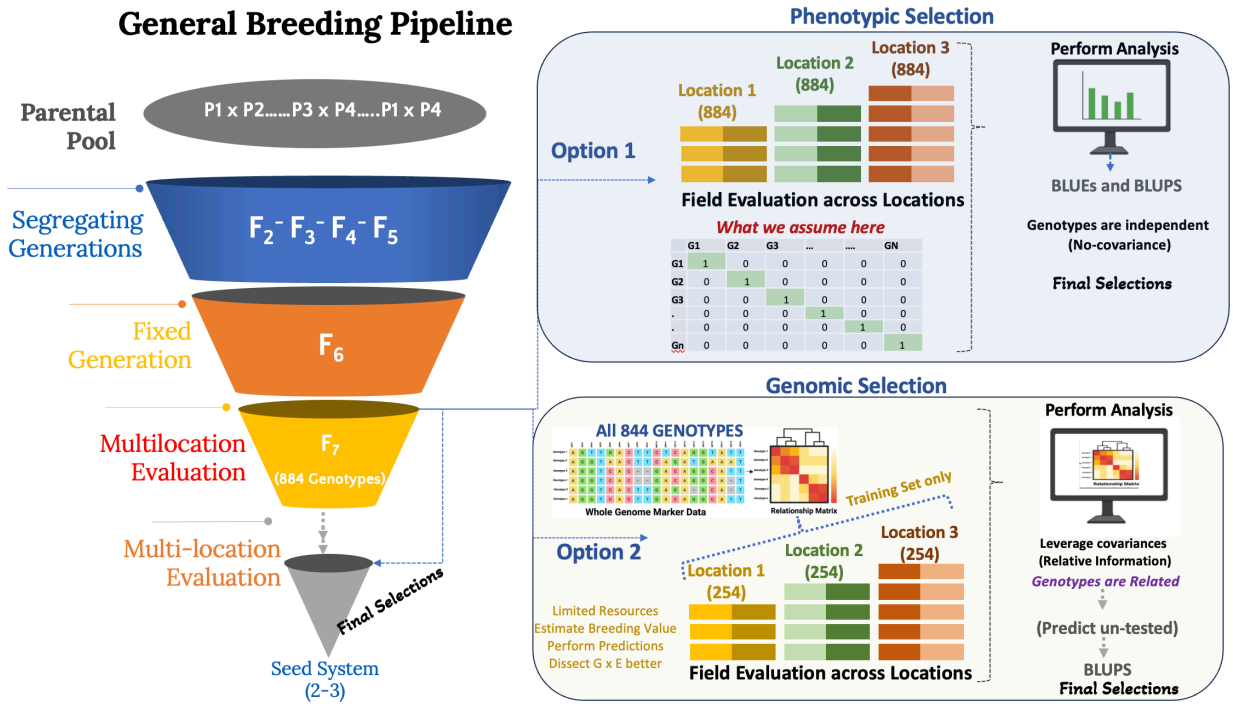
For this, we will use a multi bi-parental mapping population derived by crossing multiple parents. The population is fixed at F_7 generation. The total number of genotypes in the population are **844**. The whole population has been genotyped with the **GBS SNP data** with total number of ~396511 SNP markers.

This mapping population has been divided into training set with 252 genotypes phenotyped across multiple enviornments for Grain yield and other traits. The rest of the 592 genotypes has only genotyped but not phenotyped. We will predict the performance of the 592 genotypes and estimate breeding values for grian yield for them.

What is Our Goal

- **Part 1**: Perform Genomic Prediction/Selection and Predict the performance of un-tested 592 genotypes and Estimate the Breeding Values
- **Part 2**: Disect the G x E interactions and Estimate the Breeding Values

See the Figure Below for More Representation

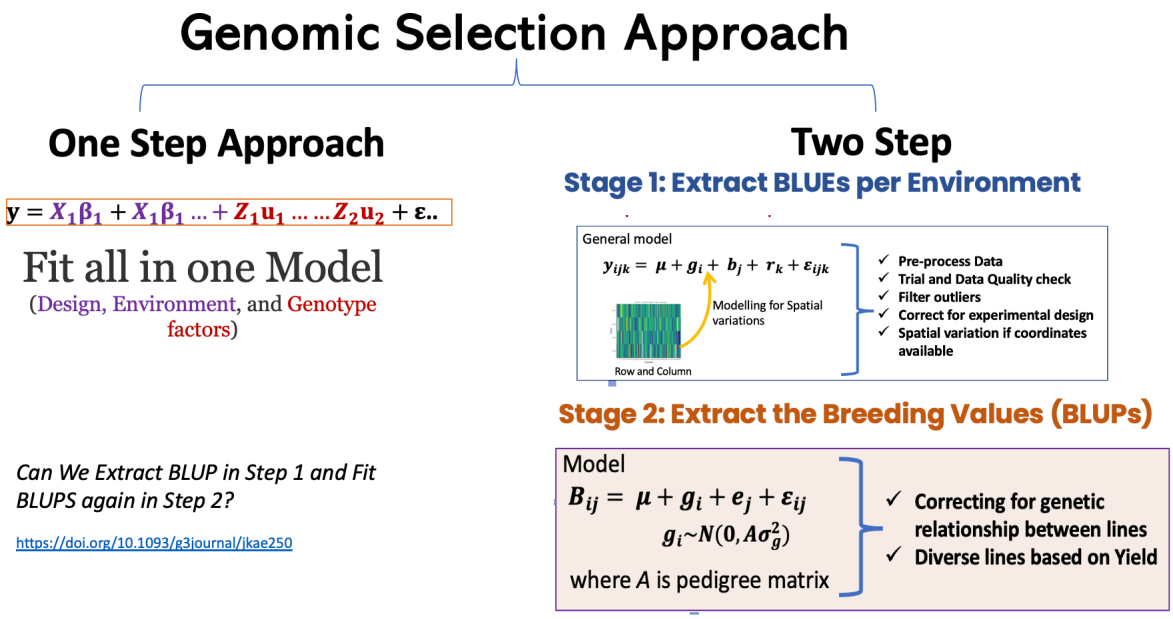


Process of Using Genomic Selection

Genomic selection in breeding or any population can be performed using either a) Single Step or 2) Two Step Approach.

Here will show example of **Two-step Genomic selection**, in which non-genetic effects and genetic effects are modeled separately. More details on on single-step and two-step GS can be found here: Resource 1, Resource 2, Resource 3, Resource 4.

In Single Step approach design, environmental and markers as covariance are fitted in one model at one time. This is not ideal and may be computationally challenging.



Step 1 Analysis

In first step we will pre-process the data, remove bad data points and then fit the model to extract the BLUEs per environment by accounting for factorial design factors, either replications, blocks, rows or columns. We will perform Single Trial analysis using an open source R package called **lme4**.

Load the Libraries

```
> # Load the Required Libraries
>   rm(list=ls()) # Remove previous work
>   library(rrBLUP)
>   library(BGLR)
>   library(AGHmatrix)
>   #library(cuTools)
>   library(dplyr)
>   library(arm)
>   library(ggplot2)
>   library(DT)
>   #library(cuTools)
>   library(lme4)
>   library(reshape2)
>   library(data.table)
>   library(sommer)
```

This section shows the analysis of filtered phenotypic data in lme4 and other open source R packages. The filtered data set was obtained after pre-processing and Quality check of data

Upload the Phenotypic Data

The Data has 5 environments and has yield data. The data comes from the different locations in Bangladesh and India.

```
> # Read the phenotypic data
>   pheno<-read.csv(file="./Data/RAW.DATA1.csv", header=TRUE)
>   str(pheno)
```

```
'data.frame':   2580 obs. of  8 variables:
 $ Environment: chr  "ENV1" "ENV1" "ENV1" "ENV1" ...
 $ Season     : chr  "2018DS" "2018DS" "2018DS" "2018DS" ...
 $ Replication: int  1 2 1 2 1 2 1 2 1 2 ...
 $ Block      : logi  NA NA NA NA NA NA ...
 $ Row        : int  6 4 4 6 6 1 4 3 3 5 ...
 $ Col        : int  15 77 1 80 6 62 13 60 1 62 ...
 $ Yield      : num  1507 1500 980 NA 2653 ...
 $ Genotype   : chr  "Genotype_10162" "Genotype_10162" "Genotype_10164" "Genotype_10164" ...
```

```
>   pheno$Environment<-as.factor(pheno$Environment) # Convert environment as factor
>   pheno$Season<-as.factor(pheno$Season) # Convert Season as factor
>   pheno$Genotype<-as.factor(pheno$Genotype) # Convert Genotype as factor
>   pheno$Block<-as.factor(pheno$Block) # Convert Block as factor
>   pheno$Replication<-as.factor(pheno$Replication) # Convert Replcation as factor
>   #table(pheno$Environment)
```

Show Phenotypic Data as Table

```
>  kable(head(pheno))
```

Environment	Season	Replication	Block	Row	Col	Yield	Genotype
ENV1	2018DS	1	NA	6	15	1507.4806	Genotype_10162
ENV1	2018DS	2	NA	4	77	1500.0775	Genotype_10162
ENV1	2018DS	1	NA	4	1	980.0517	Genotype_10164
ENV1	2018DS	2	NA	6	80	NA	Genotype_10164
ENV1	2018DS	1	NA	6	6	2653.0814	Genotype_10169
ENV1	2018DS	2	NA	1	62	NA	Genotype_10169

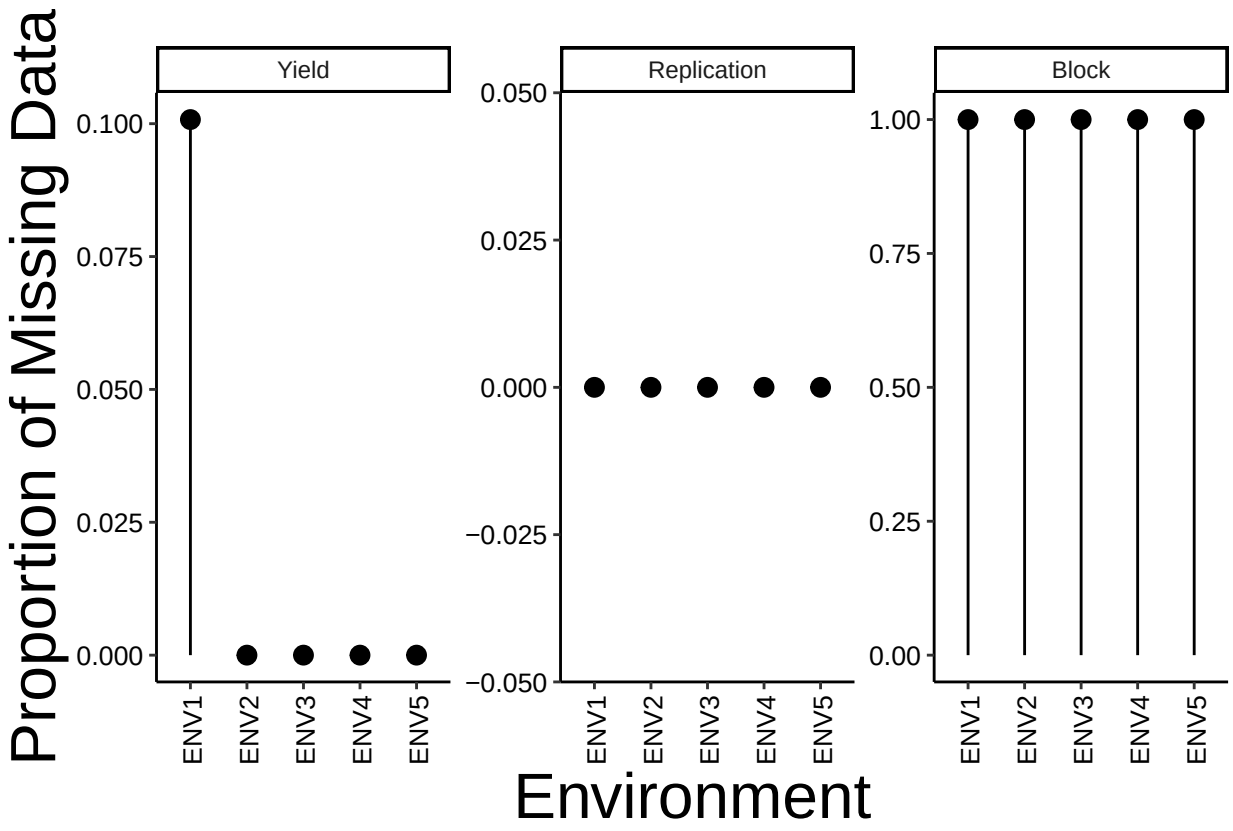
```
>  str(pheno)
```

```
'data.frame':   2580 obs. of  8 variables:
 $ Environment: Factor w/ 5 levels "ENV1","ENV2",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ Season     : Factor w/ 1 level "2018DS": 1 1 1 1 1 1 1 1 1 1 ...
 $ Replication: Factor w/ 2 levels "1","2": 1 2 1 2 1 2 1 2 1 2 ...
 $ Block      : Factor w/ 0 levels: NA NA NA NA NA NA NA NA NA NA ...
 $ Row        : int  6 4 4 6 6 1 4 3 3 5 ...
 $ Col        : int  15 77 1 80 6 62 13 60 1 62 ...
 $ Yield      : num  1507 1500 980 NA 2653 ...
 $ Genotype   : Factor w/ 259 levels "Check1","Check2",..: 8 8 9 9 10 10 11 11 12 12 ...
```

Check for Missing Data

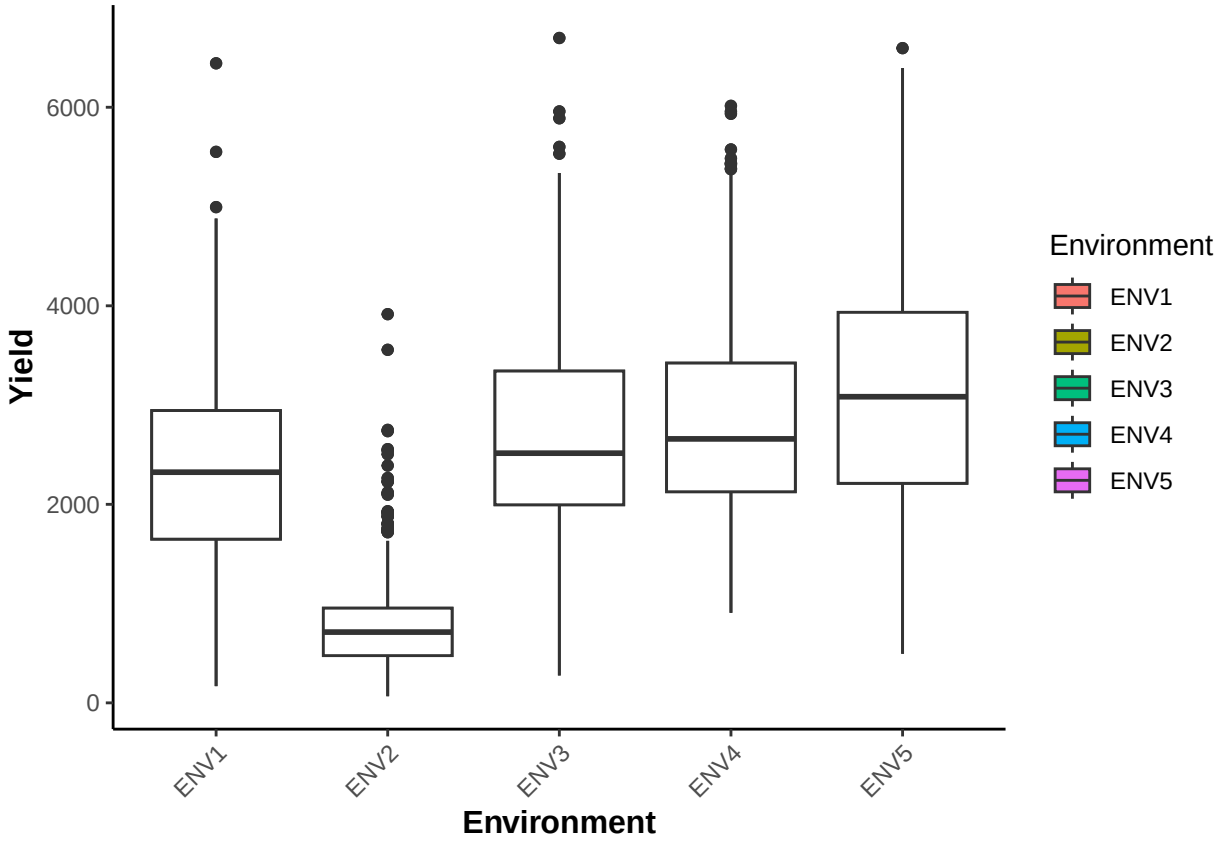
```
> # Check the missing data in each location
>    Data.missing<-data.frame(pheno %>%group_by(Environment) %>%
+                          summarise_each(funs(sum(is.na(.))/length(.))))
>   # Extract the three variables
```

```
>   Data.missing<-Data.missing[, c("Environment", "Yield", "Replication", "Block")]
>   Data.missing<-melt(setDT(Data.missing), id.vars = c("Environment"),
+                      variable.name = "Trait")
>
> # Plot the missing plot for Grain Yield
>    #png(file = "./Outputs/Plots/Missing.data.png", width =12,
>     # height =6, units = "in", res = 600)
>     ggplot(Data.missing, aes(x=Environment, y=value))+
+      geom_point(size=3) +
+      geom_segment(aes(x=Environment,
+                               xend=Environment,
+                               y=0,
+                              yend=value)) +
+       labs(title="", y="Proportion of Missing Data", x="Environment" )+
+      theme_classic()+
+      theme(axis.text.x = element_text(angle=90, vjust=0.6))+
+      #gghighlight(max(value) > .05, label_key =Environment)+
+      facet_wrap(~Trait , ncol = 3,nrow=1,scales = "free")+
+      theme (plot.title = element_text(color="black", size=14, hjust=0.5),
+                   axis.title.x = element_text(color="black", size=24),
+                   axis.title.y = element_text(color="black", size=24))+
+      theme(axis.text= element_text(color = "black", size = 10))
```



Visualize as Boxplot

```
> # Get the Box plot
>   ggplot(pheno, aes(x=Environment, y=Yield))+
+     geom_boxplot(aes(fill=Environment))+
+     theme_classic()+
+     geom_boxplot()+
+     theme(axis.text.x = element_text(angle = 45, hjust = 1)) +# fill by timepoint to give different c
+      #scale_fill_manual(values = c("", ""))+
+     #scale_color_manual(values = c("", ""))
+     theme (plot.title = element_text(color="black", size=12,hjust=0.5, face = "bold"), # add and modi
+           axis.title.x = element_text(color="black", size=12, face = "bold"), # add and modify title
+           axis.title.y = element_text(color="black", size=12, face="bold"))  # add and modify title t
```




```
>      #scale_y_continuous(limits=c(0,15000), breaks=seq(0,15000,1000), expand = c(0, 0)) theme(legend.
```

Summarize the Yield data

```
>   summary.Yield<-data.frame(pheno %>%
+                 group_by(Environment)%>%
+                 summarize(Mean = mean(Yield, na.rm=TRUE),
+                 Median= median(Yield, na.rm=TRUE),
+                 SD =sd(Yield, na.rm=TRUE),
+                 Min.=min(Yield, na.rm=TRUE),
+                 Max.=max(Yield, na.rm=TRUE),
+                 CV=sd(Yield, na.rm=TRUE)/mean(Yield, na.rm=TRUE)*100,
```

```
+                 St.err= sd(Yield, na.rm=TRUE)/sqrt(length(Yield))
+                 ))
>   kable(summary.Yield)
```

Environment	Mean	Median	SD	Min.	Max.	CV	St.err
ENV1	2319.8574	2323.4690	926.4865	168.21705	6442.726	39.93722	40.78629
ENV2	788.2912	712.8466	491.7371	66.78593	3915.564	62.38014	21.64752
ENV3	2750.3748	2515.0487	1021.6372	275.63953	6697.758	37.14538	44.97507
ENV4	2844.2960	2659.2394	1012.9038	906.97674	6013.953	35.61176	44.59060
ENV5	3130.7905	3083.7960	1208.5663	494.32413	6595.551	38.60259	53.20417

Fit the Model to Extract BLUEs

Here we will run the analysis and extract the BLUEs for each environment. Note within environment we have block, replications and seasons.So we have to adjust for all three factors. However, only certain environments have season and those without season data has only replications. So for the environments with season and blocks information we will use model $Yield = \mu + Genotype + Season + Block + Erorr$. And if season and block information for certain environments is absent we will use model $Yield = \mu + Genotype + Replication + Erorr$.

```
> # Use lme4 R package
>   pheno$Environment<- as.character(pheno$Environment) # Environment as character for use for loop
>   un.exp<- unique(pheno$Environment) # unique environments
>   for(i in 1:length(un.exp)){ # use for loop from 1 to n environments
+     sub<- droplevels.data.frame(pheno[which(pheno$Environment==un.exp[i]),]) # Run per environment
+     if (length(levels(sub$Season))>1){ # If season is more than one fit below model
+
+       model<-lmer(Yield ~ Genotype+ (1|Season)+ (1|Block), data=sub)
+
+     } else { # if season is just one fit then this model
+       model<-lmer(Yield ~ Genotype + (1|Replication), data=sub)
+     }
+     estimates<-data.frame(BLUEs=fixef(model)[-1], Environment=un.exp[i]) # Extract BLUEs
+     estimates$BLUEs<-estimates$BLUEs+fixef(model)[1] # Add intercept
+     estimates$Genotype<-row.names(estimates) # Add names
+     if(i>1){
+       BLUEs.all<-rbind(BLUEs.all, estimates) # combine all across environments
+     }
+     else{
+       BLUEs.all<- estimates
+     }
+   }
> # Save the BLUES out put file for Genomic Predictions
>   BLUEs.all$Genotype<-gsub("^.{8}", "",  BLUEs.all$Genotype)
>  # Save the file in csv formate
>  #write.csv( BLUEs.all, file="BLUES.ALL.csv")
```

Shows BLUEs in table

```
>   kable(head(BLUEs.all))
```

	BLUEs	Environment	Genotype
GenotypeCheck3	2878.634	ENV1	Check3
GenotypeCheck4	3586.108	ENV1	Check4
GenotypeCheck5	2707.087	ENV1	Check5
GenotypeCheck6	2370.153	ENV1	Check6
GenotypeCheck7	1508.282	ENV1	Check7
GenotypeGenotype_10162	1503.779	ENV1	Genotype_10162

We are Done with Step 1 of Extracting BLUEs.

Step 2 of Analysis

In Step 2 of analysis, we will now use the BLUES from first step to fit in the second step along with the marker data (Relationship Matrix) to extract the BLUPs or what we called Genomic Estimated Breeding Values. Here we will show different scenarios of fitting the step 2 Model

a) Scenario 1: All genotypes have Phenotype and Marker Data

Here we assume that we just have 252 genotypes which were both pehenotyed and genotyped. We will fit the gBLUP model using using GRM as covariance matrix and extract the GEBVs

Read the BLUEs

I have saved the above BLUEs in data folder, we will read it and use it

```
>   BLUEs.all<-read.csv(file="./Data/BLUES.ALL.csv")
```

Read SNP data (GBS data)

This marker data as 844 genotypes with 396511 SNP Markers, and the file is saved as **.rds**. We will subset 252 genotypes and use it estimate the **GEBVs**.

```
>   geno<-readRDS("./Data/GBS_datav2.rds")
>   dim(geno)
```

```
[1]    844 396511
```

```
> # Match genotype with Phenotype
>   Ids<-unique(BLUEs.all$Genotype)
>   length(Ids)
```

```
[1] 252
```

```
> # Now subet the genotype Data based on IDs
>   geno<-geno[row.names(geno)%in%Ids,]
>   dim(geno)
```

```
[1]    252 396511
```

Build the G matrix

- Here we will construct the **Genomic Relationship Matrix (GRM)** using marker data. The GRM will be based on **VanRanden (2008)**.
- The steps used to create this GRM is:
  - Create a center of marker data (X matrix)
  - Create a Cross Product (XX)
  - Divide the (XX) by number of markers

$$GRM = XX^t/m$$

- More on relationship matrix can be found here Source 1, Source2
- We will use the AGHmatrix package to build G matrix.

```
>   GM<- Gmatrix(SNPmatrix=geno, missingValue=NA,
+                            maf=0.05, method="VanRaden")
```

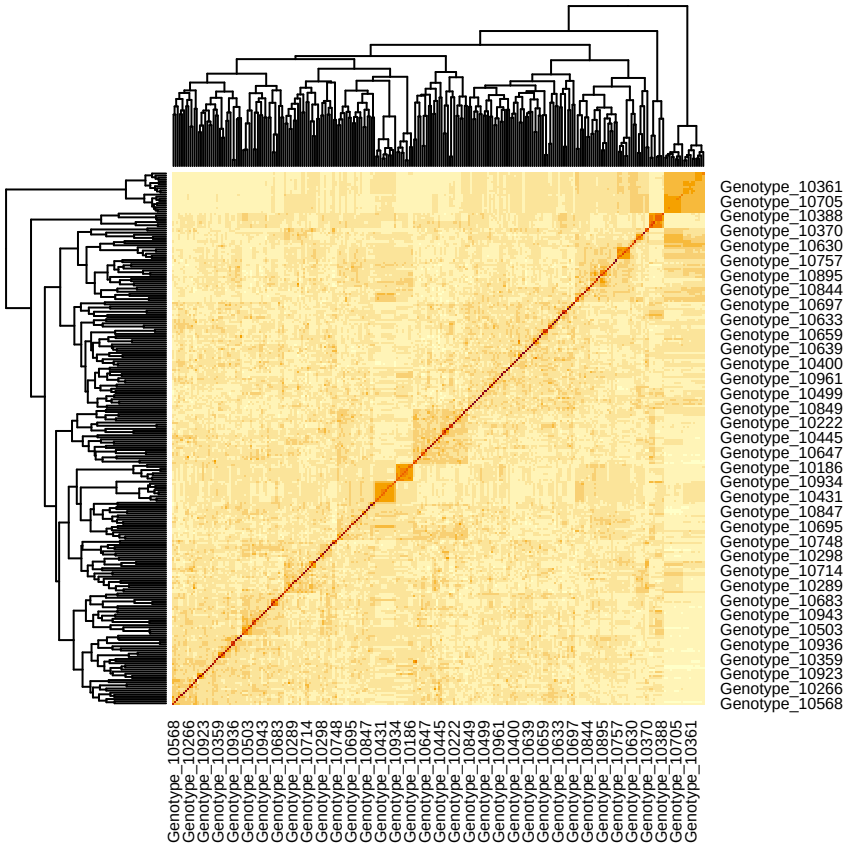
```
Initial data:
    Number of Individuals: 252
    Number of Markers: 396511

Missing data check:
    Total SNPs: 396511
     0 SNPs dropped due to missing data threshold of 0.5
    Total of: 396511  SNPs

MAF check:
     24891 SNPs dropped with MAF below 0.05
    Total: 371620  SNPs

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

Summary check:
    Initial:  396511 SNPs
    Final:  371620  SNPs ( 24891  SNPs removed)

Completed! Time = 30.976  seconds
```

```
>   dim(GM)
```

```
[1] 252 252
```

Heat Map of the GM matrix

```
>   heatmap(GM)
```



Fit the 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.

- Note we will use the **G Matrix** rather than Marker design matrix to fit the gBLUP model.
- Note in GBLUP model u represent the *Genomic Estimated Breeding Values **not marker effects. It is called** BLUP of Breeding Value**
- We will use *SOMMER* R package Click Here

```
> # Write the model
>   g_blup<- mmer(BLUEs~1, # BLUES as response variable name of column in data
+               random=~vsr(Genotype,Gu=GM)+Environment, # vsr function take covariance structure
+               rcov=~units, nIters=3,data=BLUEs.all,verbose = FALSE)
>   summary(g_blup)
```

```
$groups
            BLUEs
u:Genotype    252
Environment     5
```

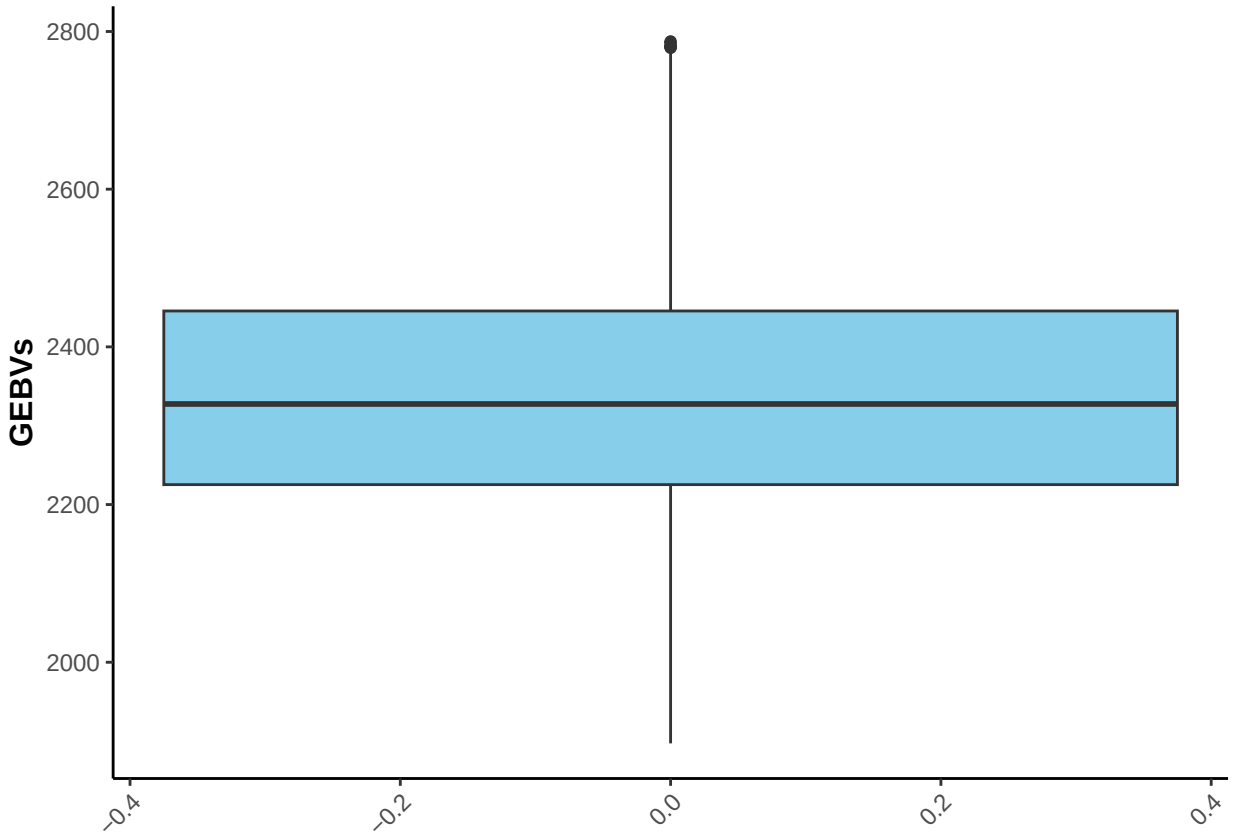
```
$varcomp
                           VarComp  VarCompSE    Zratio Constraint
u:Genotype.BLUEs-BLUEs   19965.85   7436.814  2.684731   Positive
Environment.BLUEs-BLUEs 843215.34 521266.923  1.617627   Positive
units.BLUEs-BLUEs       536007.42  22730.500 23.580978   Positive

$betas
  Trait      Effect Estimate Std.Error t.value
1 BLUEs (Intercept) 2356.097  411.1798 5.73009

$method
[1] "NR"

$logo
         logLik      AIC      BIC Method Converge
Value -127.2639 256.5277 261.6658     NR    FALSE

attr(,"class")
[1] "summary.mmer" "list"
```

```
>   #BLUPs<-g_blup$U$`u:Genotype`$BLUEs
>  # g_blup$Beta[1,3]
>   estimated<-data.frame(GEBVs= g_blup$U$`u:Genotype`$BLUEs) # Extract the Random effects
>   estimated$GEBVs<-estimated$GEBVs+ g_blup$Beta[1,3] # Add intercept (mean)
>   kable(head( estimated)) # Show in Table
```

	GEBVs
Genotype_10162	2098.487
Genotype_10164	2472.177
Genotype_10169	2197.577
Genotype_10173	2100.585
Genotype_10175	2168.149
Genotype_10176	2214.932

Visualize GEBVs as Boxplot

```
> # Get the Box plot
>   ggplot(estimated, aes(y = GEBVs)) +
+       geom_boxplot(fill="skyblue")+
+       theme_classic()+
+       theme(axis.text.x = element_text(angle = 45, hjust = 1)) +# fill by timepoint to give different
+      #scale_fill_manual(values = c("", ""))+
+     #scale_color_manual(values = c("", ""))
+     theme (plot.title = element_text(color="black", size=12,hjust=0.5, face = "bold"), # add and modi
+           axis.title.x = element_text(color="black", size=12, face = "bold"), # add and modify title
+           axis.title.y = element_text(color="black", size=12, face="bold"))  # add and modify title t
```




```
>      #scale_y_continuous(limits=c(0,15000), breaks=seq(0,15000,1000), expand = c(0, 0))
```

b) Scenario 2:Predict Breeding Values

Here we will demonstrate the example of how to predict the performance of un-tested genotypes and estimate GEBVs for all and perform selections.

Read the marker data

```
> # Read the same marker data
>   geno_all<-readRDS("./Data/GBS_datav2.rds")
>   dim(geno_all)
```

```
[1]    844 396511
```

Build the G matrix (Big matrix of 844 x 844)

```
>   GM_all<- Gmatrix(SNPmatrix=geno_all, missingValue=NA,
+                            maf=0.05, method="VanRaden")
```

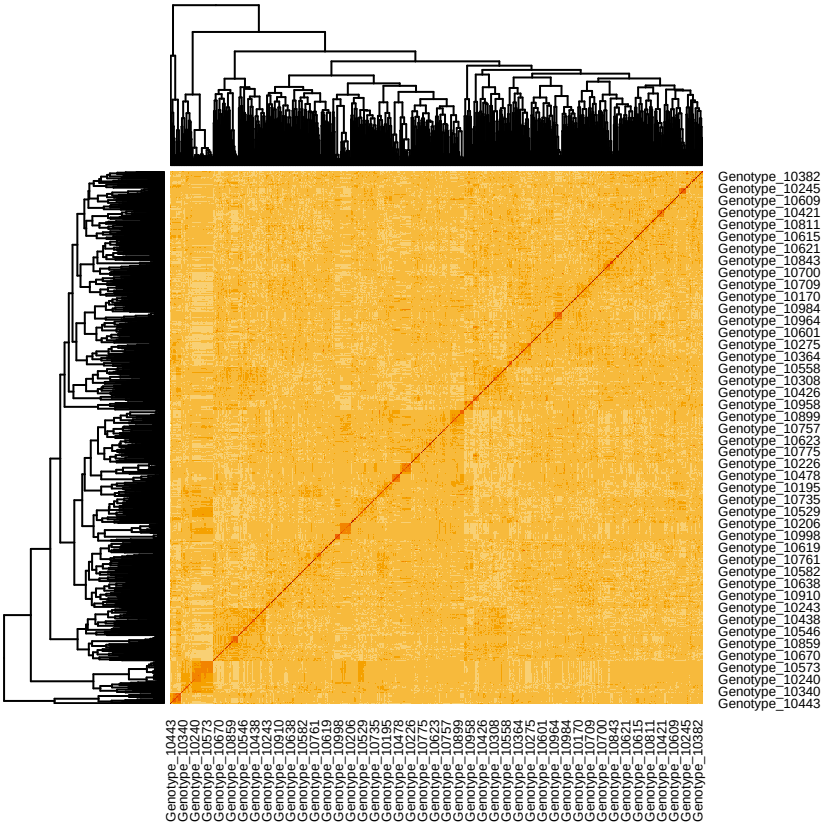
```
Initial data:
    Number of Individuals: 844
    Number of Markers: 396511

Missing data check:
```

```
    Total SNPs: 396511
     0 SNPs dropped due to missing data threshold of 0.5
    Total of: 396511  SNPs

MAF check:
    No SNPs with MAF below 0.05

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

Summary check:
    Initial:  396511 SNPs
    Final:  396511  SNPs ( 0  SNPs removed)

Completed! Time = 136.358  seconds
```

```
>    dim(GM_all)
```

```
[1] 844 844
```

```
>    heatmap(GM_all)
```



Build the gBLUP Model and Predict

```
>    gs.model1<- mmer(BLUEs~1,
+                 random=~vsr(Genotype,Gu=GM_all)+Environment,
+                 rcov=~units, nIters=3,data=BLUEs.all,verbose = FALSE)
```

```
Adding additional levels of Gu in the model matrix of 'Genotype'
```

```
>   summary(gs.model1)
```

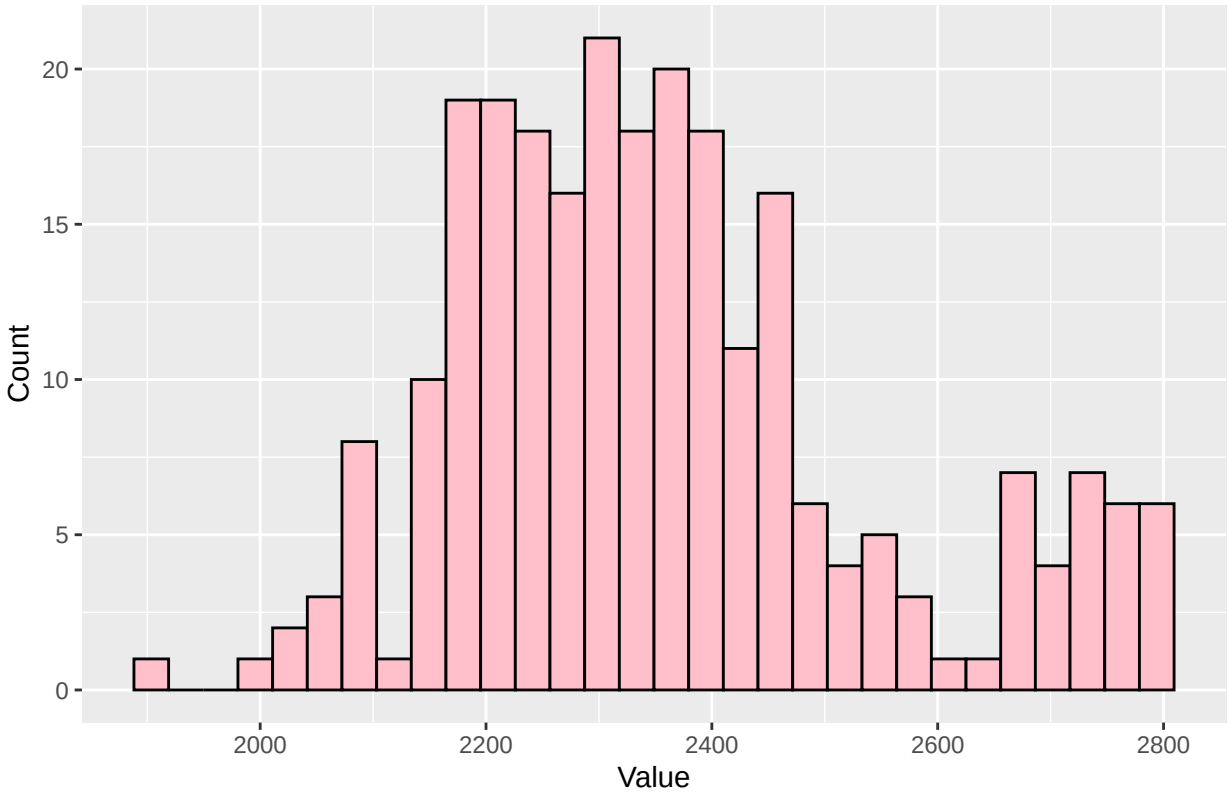
```
$groups
            BLUEs
u:Genotype    844
Environment     5

$varcomp
                           VarComp  VarCompSE    Zratio Constraint
u:Genotype.BLUEs-BLUEs   20734.36   7638.793  2.714350   Positive
Environment.BLUEs-BLUEs 843246.78 521484.161  1.617013   Positive
units.BLUEs-BLUEs       535842.21  22745.282 23.558389   Positive

$betas
  Trait      Effect Estimate Std.Error  t.value
1 BLUEs (Intercept)  2323.85  411.3257 5.649661

$method
[1] "NR"

$logo
         logLik      AIC      BIC Method Converge
Value -127.3217 256.6434 261.7815     NR    FALSE

attr(,"class")
[1] "summary.mmer" "list"
```

```
>   #gs.model1$U$`u:Genotype`$BLUEs
>   #gs.model1$Beta[1,3]
>   estimated.all<-data.frame(GEBVs= gs.model1$U$`u:Genotype`$BLUEs)
>   estimated.all$GEBVs<-estimated.all$GEBVs+ gs.model1$Beta[1,3]
>    kable(head(estimated.all)) # Show in Table
```

	GEBVs
Genotype_10162	2095.309
Genotype_10164	2472.368
Genotype_10169	2192.822
Genotype_10173	2099.613
Genotype_10175	2167.382
Genotype_10176	2216.615

Visualize the GEBVs

```
> ggplot(data=estimated, aes(GEBVs))+
+           #geom_density(alpha = 0.5)+
+           geom_histogram(fill="pink", color="black")+
+           #theme_few()+ #use white theme
+           labs(title="",x="Value", y = "Count")
```



Extract the Additional Components

Variance Componnets

```
> #var comps
>   sm <- summary( gs.model1)$varcomp
>   sm
```

	VarComp	VarCompSE	Zratio	Constraint
u:Genotype.BLUEs-BLUEs	20734.36	7638.793	2.714351	Positive
Environment.BLUEs-BLUEs	843246.78	521484.161	1.617013	Positive
units.BLUEs-BLUEs	535842.21	22745.282	23.558389	Positive

Heritability

```
>   library(sommer)
>   vg <- sm[grepl('Genotype', row.names(sm)), 1]
>   ve <- sm[grepl('units', row.names(sm)), 1]
>   #hertability <- vpredict(gs.model1, h2 ~ V1 / (V1 + V2))
>   #hertability
>   #h2 <- vpredict(gs.model1,dam.prop~V1/ (V1 + V2+V3))
>   #h2
>   #h2<-vg/(vg+ve)
>   #h2
```

Reliability (Prediction Accuracy)

```
> # Get prediction Error Variance
>   pev <- diag(gs.model1$PevU$`u:Genotype`$BLUEs)
> # Get Reliability
>   reliability<- data.frame(r2=1 - pev / vg) # Recall the formulla of reliability
>   head(reliability)
```

	r2
Genotype_10162	-0.2857817
Genotype_10164	0.2085360
Genotype_10169	-0.3068584
Genotype_10173	-0.1838010
Genotype_10175	-0.3573984
Genotype_10176	-0.3555522

Ranking of GEBVs

Here we will rank the genotypes based on high GEBVs and we will also take reliability values in consideration

```
>   estimated.all$Genotype<-rownames(estimated.all) # Assign column to genotypes
>   reliability$Genotype<-rownames(reliability) # Assign names to column
>   GEBVs.all<-merge(estimated.all, reliability, by="Genotype")
> # Arrange the BLUPs in decreasing order
>   GEBVs.all<-GEBVs.all%>%arrange(desc(GEBVs))
> # Now select Top 40
>   GEBVs.top50<-data.frame(GEBVs.all[1:50, ])
>   kable( GEBVs.top50)
```

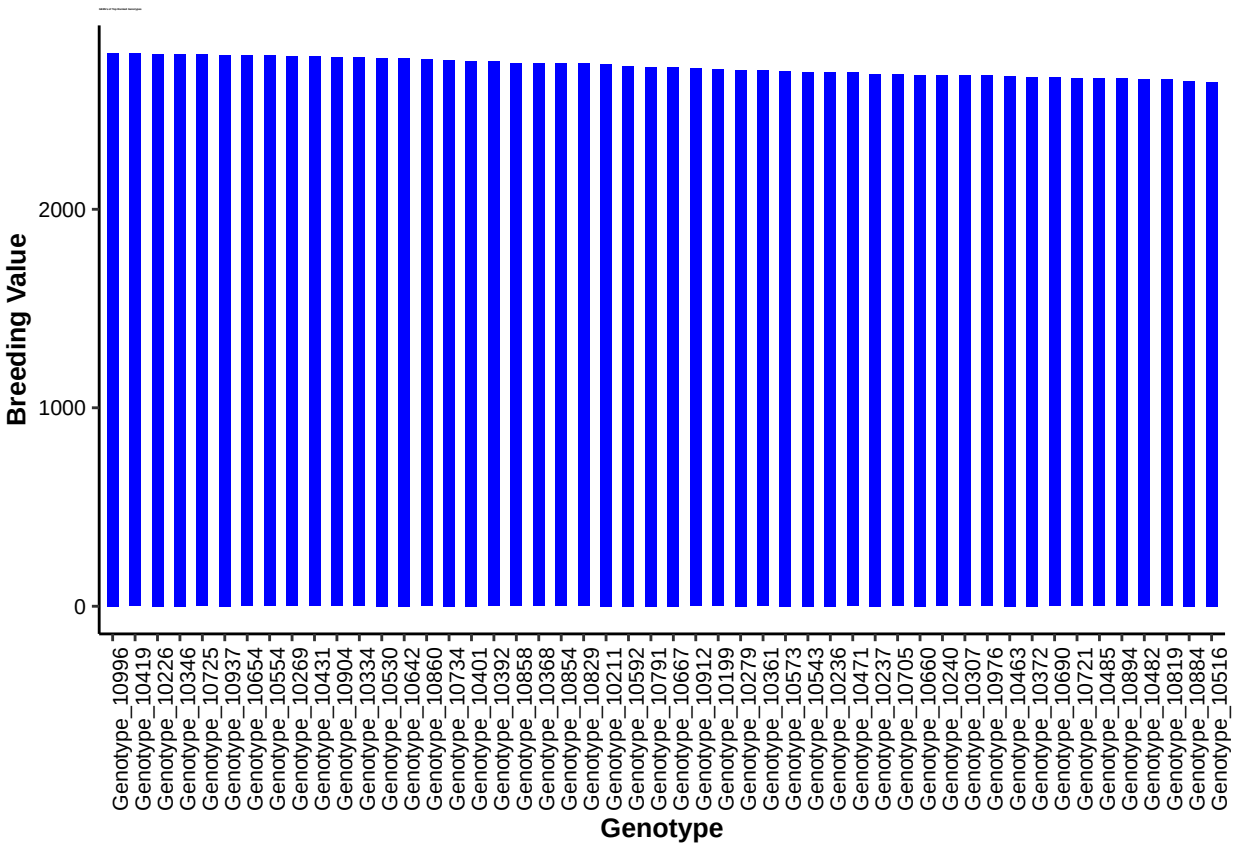
Genotype	GEBVs	r2
Genotype_10996	2787.273	0.4830401
Genotype_10419	2784.877	0.4106330
Genotype_10226	2781.540	0.3113667
Genotype_10346	2781.522	0.3685744
Genotype_10725	2779.262	0.4922281
Genotype_10937	2777.866	0.3543060
Genotype_10654	2775.064	0.4747539
Genotype_10554	2774.572	0.4004590
Genotype_10269	2770.966	0.1352096
Genotype_10431	2770.821	0.3247684
Genotype_10904	2764.602	0.4791761
Genotype_10334	2763.889	0.3431836
Genotype_10530	2762.532	0.3907696
Genotype_10642	2762.370	0.3587218
Genotype_10860	2754.928	0.2562921
Genotype_10734	2752.234	0.2482006
Genotype_10401	2746.322	0.4125898
Genotype_10392	2745.546	0.2968989
Genotype_10858	2735.536	0.2498909
Genotype_10368	2734.873	0.3334232
Genotype_10854	2734.861	0.4754298

Genotype	GEBVs	r2
Genotype_10829	2734.065	0.3126015
Genotype_10211	2732.648	0.4166438
Genotype_10592	2721.024	0.2894591
Genotype_10791	2717.661	0.2339575
Genotype_10667	2716.685	0.3447847
Genotype_10912	2710.278	0.0130428
Genotype_10199	2704.747	0.2068743
Genotype_10279	2701.463	0.4324647
Genotype_10361	2699.963	0.4771070
Genotype_10573	2697.181	0.3110623
Genotype_10543	2692.941	0.4369726
Genotype_10236	2692.522	0.3332725
Genotype_10471	2690.100	0.1407331
Genotype_10237	2682.218	0.3558803
Genotype_10705	2678.144	0.5492017
Genotype_10660	2676.731	0.3749078
Genotype_10240	2674.758	0.3825149
Genotype_10307	2674.545	0.3521213
Genotype_10976	2673.408	-0.0662195
Genotype_10463	2671.940	0.4567652
Genotype_10372	2666.804	0.4612747
Genotype_10690	2665.173	0.4032171
Genotype_10721	2660.212	0.5057958
Genotype_10485	2657.897	0.0709214
Genotype_10894	2657.803	0.4061087
Genotype_10482	2654.269	0.3105311
Genotype_10819	2653.366	0.3598400
Genotype_10884	2646.702	0.3134695
Genotype_10516	2642.572	0.3864092

```
>   #write.csv(GEBVs.all, file="./Data/GEBVs.all.csv")
```

Visualize top genotypes with high GEBVs

```
> # Miantain order from top to bottom
>   GEBVs.top50$Genotype <- factor(GEBVs.top50$Genotype,
+                                  levels=unique(GEBVs.top50$Genotype))
> # Visualize as bar plot
>   bar.plot<-ggplot(data=GEBVs.top50, aes(x=Genotype, y=GEBVs)) +
+   geom_bar(stat="identity", width=0.5, fill="blue")+
+   theme_classic()+
+   labs(title="GEBVs of Top Ranked Genotypes",x="Genotype", y = "Breeding Value")+
+   #scale_y_continuous(limits = c(0, 6000), breaks = seq(0, 6000, by = 500))+
+   theme (plot.title = element_text(color="black", size=1, face="bold", hjust=0),
+            axis.title.x = element_text(color="black", size=10, face="bold"),
+            axis.title.y = element_text(color="black", size=10, face="bold")) +
+    theme(axis.text= element_text(color = "black", size = 8))+
+   theme(axis.text.x = element_text(angle = 90, hjust = 1))
>  bar.plot
```



Next We will learn How to Fit the Higher Models and Disect G x E interactions

---

Additional Literature

- Screening experimental designs
- Analysis and Handling of G × E in a Practical Breeding Program
- A stage□wise approach for the analysis of multi□environment trials
- Analysis of series of variety trials with perennial crops
- A tutorial on the statistical analysis of factorial experiments with qualitative and quantitative treatment factor levels
- Experimental design matters for statistical analysis: how to handle blocking
- Random effects structure for confirmatory hypothesis testing: Keep it maximal
- Generalized linear mixed models: a practical guide for ecology and evolution
- Mixed Models Offer No Freedom from Degrees of Freedom
- Perils and pitfalls of mixed-effects regression models in biology
- A brief introduction to mixed effects modelling and multi-model inference in ecology
- Modeling Spatially Correlated and Heteroscedastic Errors in Ethiopian Maize Trials

- More, Larger, Simpler: How Comparable Are On□Farm and On□Station Trials for Cultivar Evaluation
- Rethinking the Analysis of Non□Normal Data in Plant and Soil Science
- The Design and Analysis of Long□Term Rotation Experiments
- Analysis of Combined Experiments Revisited
- Fundamentals of Experimental Design: Guidelines for Designing Successful Experiments

---

Note: For questions specific to data analysiss shown here contact waseem.hussain@irri.org

---

If your experiment needs a statistician, you need a better experiment - Ernest Rutherford

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