Dissecting Mixed Model Equation

Step-by-Step Guide to Extract BLUEs and BLUPs

(Dedicated to IRRI's Legacy!)

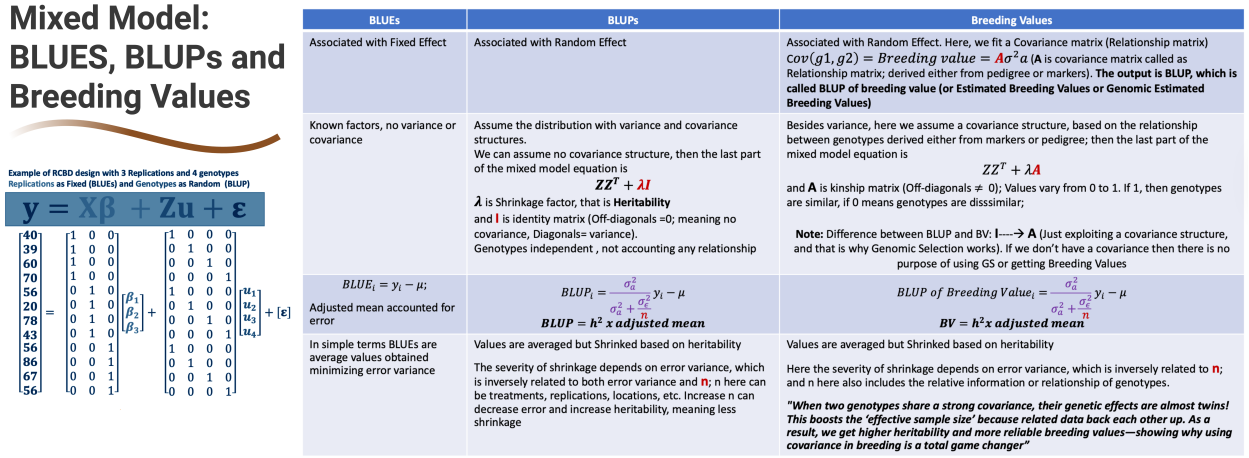
Waseem Hussain
Senior Scientist-I
International Rice Research Institute
Rice Breeding Innovations Platfrom
waseem.hussain@cgiar.org
whussain2.github.io

August 29, 2025

Contents

Introduction

In my last Linkedin Post, I introduced you to **BLUEs**, **BLUPs** and **Breeding Values**! But here's the big question: **Where do they really come from?**



Today, we're going to unlock the secret lair of the legendary mixed model equations (*Step-by-Step guide*) that is the behind driving force to give us the **BLUEs**, **BLUPs** and **Breeding Values** and more.

In this tutorial, we will learn how to extract the **BLUEs** and **BLUPs**. Next model, I will show you how to extract **Breeding values (gBLUP)!**

Here, in this tutorial I will not use any R package at first!

We'll roll up our sleeves and extract BLUEs and BLUPs by hand, piece by piece, just like old-school statisticians did before the days of fancy R packages.

For this we will manually build **Mixed Model Equation (MME)** through matrix multiplication of *Fixed* and *Random* design matrices and latter solve them to get **BLUEs** and **BLUPs**.

After we've wrestled with the raw equations and unearthed these treasures ourselves, we'll summon the power of **lme4** R package

Together, we'll compare our manually extracted BLUEs and BLUPs with those conjured by lme4 R package and see if they match — because even in a tale of numbers, double-checking is always wise

Our mystical guide on this quest will be none other than **Henderson's 1950** famous **Mixed Model Equation (MME)**.

By dissecting this scroll, we will understand exactly how mixed models work, and how BLUEs (our Best Linear Unbiased Estimates) and BLUPs (our Best Linear Unbiased Predictors) spring to life.

First Let us Recall again the Mixed Model Equation!

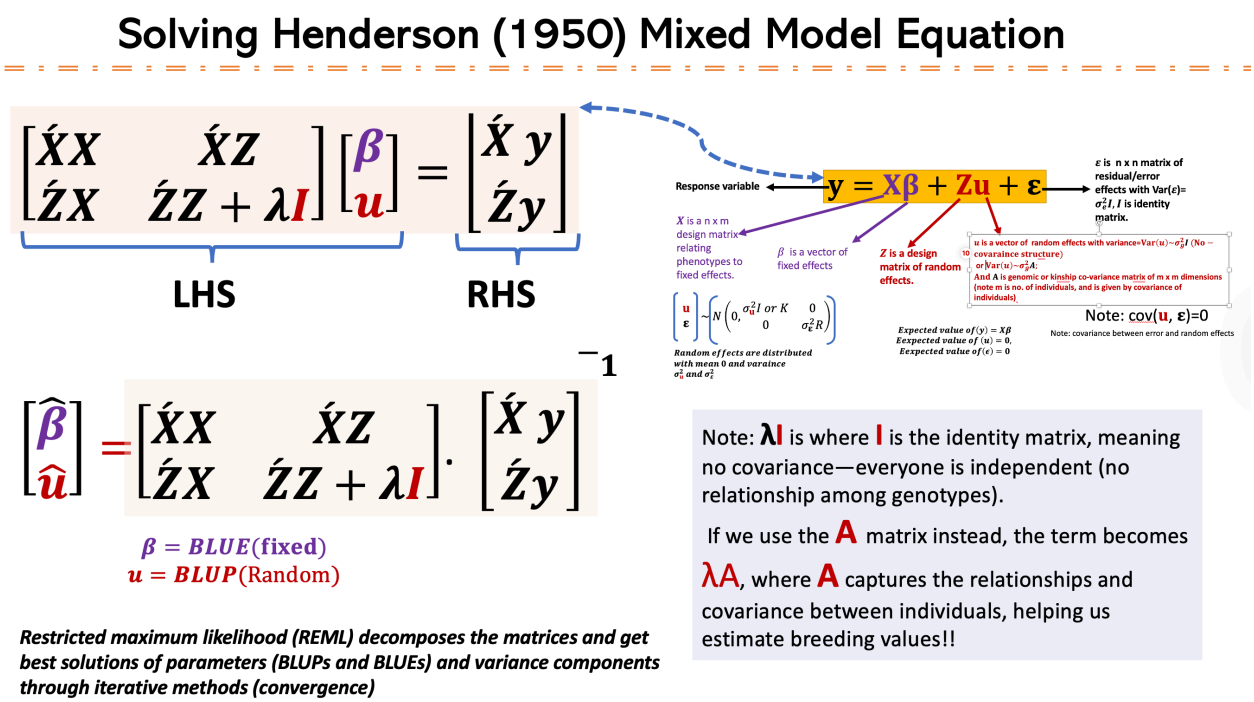
$$y = X\beta + Zu + \epsilon$$

where,

y : is the vector of observed values of the trait
X : is the incidence matrix for fixed component
β : is the fixed effect, which is environment here in demo data
Z : is the incidence matrix for random component
u : is the random effect, which is genotype here in demo data
ϵ : is the error component component

here,

$$u \sim N(0, I{\sigma_u}^2) \text{and} \epsilon \sim N(0, I{\sigma_\epsilon}^2)$$



From the mixed model equation above, one thing is crystal clear: it's really just some serious *matrix multiplication* between three key players — ***the fixed effects matrix (X)***, the ***random effects matrix (Z)***, and our ***response variable (Y)***. Think of it like a perfectly balanced seesaw, where the **Left Hand Side (LHS)** and **Right Hand Side (RHS)** must match just right.

Our mission? To solve this neat equation and unlock the magic numbers for β and u -aka the **BLUEs (fixed effects)** and the **BLUPs (random effects)** who hold the secrets to our data.

In this tutorial, we'll channel our inner math wizards and build these matrices step by step, multiply them out like a pro kitchen chef mixing ingredients, and finally solve the equations to reveal those prized solutions. So buckle up, because we're about to turn some matrices into magic!

Ready? Let's dive in — I promise, it's as fun as it sounds!

Load the required libraries

Before we dive into the matrix madness of mixed models, let's invite our trusty assistants:

```
>   library(agridat)
>   library(lme4)
```

Data upload and information

- Here we will upload the data, and looks its structure. The data has *464* observations with total *10* variables.
- For demo purpose we will use *environment (env)*, *genotype (gen)* and *yield* variables to run mixed model equation.
- Here Environment **(X)** is **Fixed Effect**, Genotype **(Z)** is **Random Effect** and Yield as Response variable **(Y)**

```
> # Load the Australia.soybean data from agridat package
>     australia.soybean<-read.csv(file="./Data/australia.soybean.csv")
>
> # Get the structure of data
>     str(australia.soybean)
'data.frame':   464 obs. of  10 variables:
 $ env    : chr  "L70" "L70" "L70" "L70" ...
 $ loc    : chr  "Lawes" "Lawes" "Lawes" "Lawes" ...
 $ year   : int  1970 1970 1970 1970 1970 1970 1970 1970 1970 1970 ...
 $ gen    : chr  "G01" "G02" "G03" "G04" ...
 $ yield  : num  2.39 2.28 2.57 2.88 2.39 ...
 $ height : num  1.45 1.45 1.46 1.26 1.33 ...
 $ lodging: num  4.25 4.25 3.75 3.5 3.5 4 3 3.25 3 3.75 ...
 $ size   : num  8.45 9.95 10.85 10.05 11 ...
 $ protein: num  36.7 37.5 37.8 38.5 37.5 ...
 $ oil    : num  20.9 20.7 21.3 22 22.1 ...
> # Convert env, loc and year into factors
>     australia.soybean$env<-as.factor(australia.soybean$env)
>     australia.soybean$loc<-as.factor(australia.soybean$loc)
>     australia.soybean$year<-as.factor(australia.soybean$year)
>     australia.soybean$gen<-as.factor(australia.soybean$gen)
> # Data can also be upload directly from agridat package
>     data(australia.soybean, package = "agridat")
>     head(australia.soybean)
```

env	loc	year	gen	yield	height	lodging	size	protein	oil
L70	Lawes	1970	G01	2.387	1.445	4.25	8.45	36.70	20.895
L70	Lawes	1970	G02	2.282	1.450	4.25	9.95	37.55	20.740
L70	Lawes	1970	G03	2.567	1.460	3.75	10.85	37.80	21.295
L70	Lawes	1970	G04	2.877	1.260	3.50	10.05	38.45	21.990
L70	Lawes	1970	G05	2.392	1.335	3.50	11.00	37.50	22.130
L70	Lawes	1970	G06	2.408	1.360	4.00	11.75	38.25	21.160

```
>     str(australia.soybean)
'data.frame':   464 obs. of  10 variables:
 $ env    : Factor w/ 8 levels "B70","B71","L70",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ loc    : Factor w/ 4 levels "Brookstead","Lawes",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ year   : int  1970 1970 1970 1970 1970 1970 1970 1970 1970 1970 ...
 $ gen    : Factor w/ 58 levels "G01","G02","G03",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ yield  : num  2.39 2.28 2.57 2.88 2.39 ...
 $ height : num  1.45 1.45 1.46 1.26 1.33 ...
```

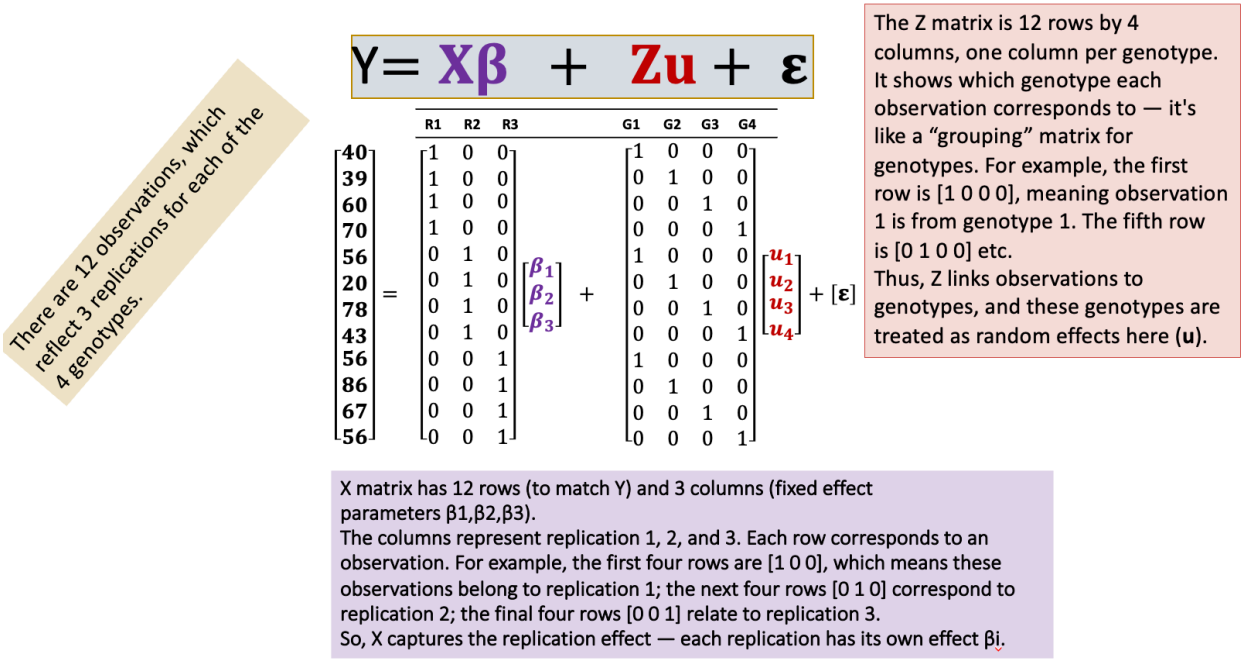
```
 $ lodging: num  4.25 4.25 3.75 3.5 3.5 4 3 3.25 3 3.75 ...
 $ size   : num  8.45 9.95 10.85 10.05 11 ...
 $ protein: num  36.7 37.5 37.8 38.5 37.5 ...
 $ oil    : num  20.9 20.7 21.3 22 22.1 ...
> # Subset the data (environment, genotypes and yield) now for mixed model equation
>     demo.data<-australia.soybean[, c(1,4,5)]
> # Look for number of environments.
>     table(demo.data$env)

B70 B71 L70 L71 N70 N71 R70 R71
 58  58  58  58  58  58  58  58
> # Look for number of environments.
>     table(demo.data$gen)

G01 G02 G03 G04 G05 G06 G07 G08 G09 G10 G11 G12 G13 G14 G15 G16 G17 G18 G19 G20
  8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8
G21 G22 G23 G24 G25 G26 G27 G28 G29 G30 G31 G32 G33 G34 G35 G36 G37 G38 G39 G40
  8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8
G41 G42 G43 G44 G45 G46 G47 G48 G49 G50 G51 G52 G53 G54 G55 G56 G57 G58
  8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8
```

Mixed model equation and build components

Here we will build the individual components of **Henderson(1950)** mixed model equation.

I am showing here one example what we mean by Design/incidence matrices for Fixed and Random effects. For Example if we have a RCBD design experiment with 4 Genotypes Replicated 3 times. Then what will be its X and Z design matrices

See Figure below



Similar way we will build the matrices for Fixed (environment as X) and Random (Genotypes as Z).

Create X Design Matrix

Here, **X** is our fearless design matrix representing the **fixed effects**—in this case, the mighty **environment (env)**.

Think of **X** as the stage setup for our fixed effect actors to perform their roles.

To build this stage perfectly, we call upon the R wizard *model.matrix*.

```
> # Building X matrix for fixed component, i.e., for environments
>     X <- model.matrix(~+env, demo.data) # +env means I am adding In
>     head(X)
  (Intercept) envB71 envL70 envL71 envN70 envN71 envR70 envR71
1           1      0      1      0      0      0      0      0
2           1      0      1      0      0      0      0      0
3           1      0      1      0      0      0      0      0
4           1      0      1      0      0      0      0      0
5           1      0      1      0      0      0      0      0
6           1      0      1      0      0      0      0      0
```

Note: in the formula +env I tell R to include an intercept (the baseline value) plus the variable env as a fixed effect in the matrix.The resulting matrix X will have one column for the intercept (usually a column of 1s), and additional columns representing each level of the factor env (coded as dummy variables)

Create Z Design Matrix

Now, meet **Z**, the design matrix for the random effects—the wild cards of our model.

In this example, the wild cards are the **genotypes**. To build this rebel squad's matrix, we again call upon the trusty *model.matrix* function, but this time with a little twist: **-1** to remove the intercept because, well, we don't want any

fixed baseline stealing the spotlight from our random heroes.

```
> # Building Z matrix for random component, i.e., for genotypes
>     Z <- model.matrix(~-1 + gen, demo.data) # -1 here is for removing intercept
```

Creat Respone Variable

Now, let's meet our star of the show **Y**, the response variable.

Here, we snatch the Yield column straight from **demo.data** to play this role. Think of **Y** as the main character in this drama: it's the outcome we want to explain or predict using all those fancy fixed and random effects.

So, Y is where all the action (data) happens, and it eagerly waits for the model to explain its twists and turns!

```
> # Yield variable as response variable
>     y <-demo.data$yield
```

Build $X^T X$

Now, let's get our hands dirty and build the famous $X^T X$.

What are we doing? Just taking our design or incidence matrix **X** and multiplying it by its transpose. It's like folding a map and then unfolding it perfectly to see how all our fixed effects connect and interact.

This multiplication helps us summarize the fixed effect structure before solving the big equation. In short, $X^T X$ is the building block that helps us peg down the fixed effects in the model, setting the stage for finding those **BLUEs (our Best Linear Unbiased Estimates)**.

```
> # Cross product and transpose
>     XtX <- t(X) %*% X
>     XtX
            (Intercept) envB71 envL70 envL71 envN70 envN71 envR70 envR71
(Intercept)         464     58     58     58     58     58     58     58
envB71               58     58      0      0      0      0      0      0
envL70               58      0     58      0      0      0      0      0
envL71               58      0      0     58      0      0      0      0
envN70               58      0      0      0     58      0      0      0
envN71               58      0      0      0      0     58      0      0
envR70               58      0      0      0      0      0     58      0
envR71               58      0      0      0      0      0      0     58
```

Build $X^T Z$

Time to bring together the *fixed effects* and *random effects* in a glamorous matrix mingle! **Z** is the incidence matrix for the random effects, and **X** holds the fixed effects—it's like setting up the party where both groups get introduced.

Think of X^T by Z as the ultimate networking event where the transpose of X scans and connects with each column of Z, making sure every fixed effect knows about every random effect's presence. The result? A sparkly matrix ready to join the mixed model dance floor!

```
> # Cross product and transpose
>     XtZ <- t(X) %*% Z
>     XtZ[, 1:10]
            genG01 genG02 genG03 genG04 genG05 genG06 genG07 genG08 genG09
(Intercept)      8      8      8      8      8      8      8      8      8
envB71           1      1      1      1      1      1      1      1      1
```

```
envL70           1      1      1      1      1      1      1      1      1
envL71           1      1      1      1      1      1      1      1      1
envN70           1      1      1      1      1      1      1      1      1
envN71           1      1      1      1      1      1      1      1      1
envR70           1      1      1      1      1      1      1      1      1
envR71           1      1      1      1      1      1      1      1      1
            genG10
(Intercept)      8
envB71           1
envL70           1
envL71           1
envN70           1
envN71           1
envR70           1
envR71           1
```

Build $Z^T X$

Next up, we're mixing the fixed and random worlds with the $Z^T X$. Think of **X (fixed effects)** and **Z (random effects)** as two dance crews coming together for a perfect duet.

When we multiply X^T by Z, it's like getting the best moves from each group synced on the dance floor.

This little cross product (is the secret handshake that shows how fixed and random effects interact in our model—no fancy software needed, just classic matrix moves!

```
> # Cross product and transpose
>     ZtX <- t(Z) %*% X
>     ZtX[1:8, ]
      (Intercept) envB71 envL70 envL71 envN70 envN71 envR70 envR71
genG01           8      1      1      1      1      1      1      1
genG02           8      1      1      1      1      1      1      1
genG03           8      1      1      1      1      1      1      1
genG04           8      1      1      1      1      1      1      1
genG05           8      1      1      1      1      1      1      1
genG06           8      1      1      1      1      1      1      1
genG07           8      1      1      1      1      1      1      1
genG08           8      1      1      1      1      1      1      1
```

Build $Z^T Z$

Now, it's time for the random effects to have their moment with the $Z^T Z$. Think of Z as the random effects' VIP guest list — when we multiply Z by Z^T it's like counting how many times each VIP showed up and who they mingled with.

```
> # Cross product and transpose
>     ZtZ <- t(Z) %*% Z
>     ZtZ[1:7,1:7]
      genG01 genG02 genG03 genG04 genG05 genG06 genG07
genG01      8      0      0      0      0      0      0
genG02      0      8      0      0      0      0      0
genG03      0      0      8      0      0      0      0
genG04      0      0      0      8      0      0      0
genG05      0      0      0      0      8      0      0
```

```
genG06      0      0      0      0      0      8      0
genG07      0      0      0      0      0      0      8
```

Variance components

Now, in this tutorial we're keeping it simple and skipping the fancy **G matrix (the relationship matrix)**, assuming **G=I**- that is, each random effect is independent, no *covariance* drama here.

We create I, the *identity matrix*, basically saying, *"Hey, treat everyone equally*, no special family ties." Then we peek at λ the **shrinkage factor**, which is just the ratio of residual variance ${\sigma_\epsilon}^2$ to random effect variance ${\sigma_u}^2$, i.e is ***Heritability***!

Variance Components, but We're Skipping the Hard Part!

Alright, here's the secret: estimating variance components $({\sigma_\epsilon}^2, {\sigma_u}^2)$ is where the real magic (and math sweat) happens. Usually, fancy software or R packages use a method called **REML (Restricted Maximum Likelihood)** to hunt down these numbers through some serious number crunching.

But fear not! We're skipping that wild rollercoaster and already know the variance components; we will plugin them in the model directly!

So no **REML** drama today — we simply plug these known values in and move on. Don't worry, your computer will thank you!

To get variance components in the above model,we assume ${\sigma_u}^2$=0.18 and ${\sigma_\epsilon}^2$=0.25.

```
>     sigmau <- 0.199 # Genotype variance
>     sigmae <- 0.25 # Residual variance
>     I <- diag(ncol(Z))  # assuming G = I, No fancy markers
>     lambda <- sigmae/sigmau # shrinkage or Heritability
```

Build $X^T y$

Time to bring together our **fixed effect stage (X)** and the star of the show **(Y, the response variable)**.

Multiplying X^T by y is like passing the microphone from the stage setup to the lead performer—getting the perfect harmony so the model can hear the full story.

This gives us a sneak peek at how the fixed effects "talk" to the response, setting the scene for those BLUEs to emerge. **Matrix multiplication has never sounded so good!**

```
> # Cross product and transpose
>     Xty <- t(X) %*% y
>     Xty
                [,1]
(Intercept) 950.006
envB71      142.478
envL70      129.687
envL71      145.687
envN70      109.483
envN71      133.362
envR70       95.098
envR71      103.862
```

Build $Z^T y$

Now it's time for our random effects matrix **Z** to have a chat with the response variable **y**.

Multiplying Z^T by **y** is like sending a smoke signal from each random effect to see how it relates to the observed data.

Think of it as the random effects checking who's yelling loudest on stage. This little matrix hears all the whispers and shouts from **y** connected to each genotype or group, getting ready for its big prediction moment!

```
> # Cross product and transpose
>     Zty <- t(Z) %*% y
>     head(Zty)
          [,1]
genG01 15.304
genG02 16.155
genG03 16.874
genG04 19.550
genG05 17.981
genG06 16.930
```

Now build LHS Equation

Welcome to the magic moment! We're assembling the puzzle pieces that make Henderson's mixed model equation come alive. The **Left Hand Side (LHS)** is where all the matrix powerhouses (fixed and random effects intersections) join forces—think of it as the Avengers assembling.

We bravely stick $X^T X$ and $X^T Z$ side by side (because they are in two columns of mixed model equation).Then stack ZTX and the cool kid $ZTZ + I * \lambda$ right underneath. Then finally *rbind* them!

```
> # Left hand side
>     LHS1 <- cbind(XtX, XtZ)
>     LHS2 <- cbind(ZtX, ZtZ + I * lambda)
>     LHS <- rbind(LHS1, LHS2)
```

Now build RHS Equation

On the other side, the **Right Hand Side (RHS)** swoops in with the response variables dancing with fixed and random matrices (bind them by rows)

```
> # Right Hand side
>     RHS <- rbind(Xty, Zty)
```

Together binding them along columns and rows as per mixed model equation is like they form the dynamic duo ready to be solved for BLUEs and BLUPs—the grand finale of our matrix adventure. Curtains up!

Solve the Equation — The Grand Finale!

Now for the magic moment: solving the **Mixed Model Equation (MME)** to get our **BLUEs (Best Linear Unbiased Estimates)** and **BLUPs (Best Linear Unbiased Predictors)**.

Think of this as cracking the code after assembling a giant Lego set. We tell R use function *solve* to do trick!

```
> # Sol function to get solution
>     sol <- solve(LHS, RHS)
```

```
>     dim(sol)
[1] 66  1
```

And voila! The first few rows are the *BLUEs—our fixed effects* heroes shining bright for each environment:

```
> # First eight are fixed effects (BLUEs) for environments
>   Blues.env<-data.frame(Blues.env=sol[1:8, ])
>   Blues.env
```

	Blues.env
(Intercept)	1.5577414
envB71	0.8987759
envL70	0.6782414
envL71	0.9541034
envN70	0.3298966
envN71	0.7416034
envR70	0.0818793
envR71	0.2329828

```
>   str(Blues.env)
'data.frame':   8 obs. of  1 variable:
 $ Blues.env: num  1.558 0.899 0.678 0.954 0.33 ...
```

Then, from row 9 onward, the *BLUPs* enter the stage, showing the *random effects*' charm—like genotypes whispering their secrets to the model:

```
> # Nine to rest are eight are random effects (BLUPs)
> # BLUP
>   Blups.gy<-sol[9:66, ]
>   head(Blups.gy)
     genG01      genG02      genG03      genG04      genG05      genG06
-0.11618205 -0.02424449  0.05343249  0.34253347  0.17302696  0.05948244
```

To top it off, we add the intercept to the BLUPs to get the final breeding values—the phenotypic BLUPs ready for the big show:

Why Add the Intercept to BLUPs?

The intercept represents the overall average or baseline level of the trait. *BLUPs* estimate the deviation of each genotype from this average (they are effects either plus or minus). Adding the intercept back to BLUPs gives the final values on the original scale, combining both the overall mean and the genotype-specific effect.

In short, without adding the intercept, the breeding values would be centered around zero and miss the real-world average. Adding it ensures the values reflect actual predicted performance.

```
> # Final genotypic values/breeding values for grain yield
>     blup.final<-data.frame(Yield.blups=Blups.gy+sol[1,1])
```

And just like that, you've magically turned matrices into meaningful estimates and predictions. Cue the applause! □□

Comparing Our Matrix Magic with lme4

Alright, now that we've hand-crafted our matrix masterpiece and summoned **BLUEs** and **BLUPs** from the depths of linear algebra, it's time for the ultimate showdown with the R superstar: **lme4 R Package* which is handy package

to fit the mixed models.

Using **lme4** is like ordering a fancy coffee instead of roasting your own beans—you get the same boost but with way less sweat!

Fit Mixed Model with lme4 Package

Here we will use **environments** as ***fixed effect*** and **genotypes** as ***random effect***, same what we did above.

```
> # Fit mixed model
>     mixemodel.fit<-lmer(yield ~ env + (1 | gen), data = demo.data)
>     summary(mixemodel.fit)
Linear mixed model fit by REML ['lmerMod']
Formula: yield ~ env + (1 | gen)
   Data: demo.data

REML criterion at convergence: 809.7

Scaled residuals:
    Min      1Q  Median      3Q     Max
-2.3272 -0.6881 -0.0006  0.5947  3.6310

Random effects:
 Groups   Name        Variance Std.Dev.
 gen      (Intercept) 0.1991   0.4462
 Residual             0.2509   0.5009
Number of obs: 464, groups:  gen, 58

Fixed effects:
            Estimate Std. Error t value
(Intercept)  1.55774    0.08808  17.686
envB71       0.89878    0.09301   9.663
envL70       0.67824    0.09301   7.292
envL71       0.95410    0.09301  10.258
envN70       0.32990    0.09301   3.547
envN71       0.74160    0.09301   7.973
envR70       0.08188    0.09301   0.880
envR71       0.23298    0.09301   2.505

Correlation of Fixed Effects:
       (Intr) envB71 envL70 envL71 envN70 envN71 envR70
envB71 -0.528
envL70 -0.528  0.500
envL71 -0.528  0.500  0.500
envN70 -0.528  0.500  0.500  0.500
envN71 -0.528  0.500  0.500  0.500  0.500
envR70 -0.528  0.500  0.500  0.500  0.500  0.500
envR71 -0.528  0.500  0.500  0.500  0.500  0.500  0.500
> # Now extract the fixed effects (BLUEs) for environmenet
>     Blues.env.lme4<-data.frame(Blues.env=mixemodel.fit@beta)
>     Blues.env.lme4
```

Blues.env
1.5577414

Blues.env
0.8987759
0.6782414
0.9541034
0.3298966
0.7416034
0.0818793
0.2329828

```
> # Now extract the random effects (BLUPs)
>     Blups.gy.lme4<-ranef(mixemodel.fit)$gen
> # Now extract the intercept and add it to random effects
>     intercept <- fixef(mixemodel.fit)[1]
> # Now add intercept to blups to get genotypic values
>     Final_blups.gy.lme4<-data.frame(Yield.blup=intercept+ranef(mixemodel.fit)$gen)
```

Now check the results of MME we solve with lme4

Then, we do a side-by-side face-off! We will round the values first and check whether two are equal.

```
> # Blues from both
>     table(round(Blues.env$Blues.env,2)==round(Blues.env.lme4$Blues.env,2))

TRUE
   8
> # Now for Blups
>     table(round(blup.final$Yield.blups, 2)==round( Final_blups.gy.lme4$X.Intercept., 2))

TRUE
  58
```

And guess what? They match perfectly! Our homemade math wizardry plays nice with the polished lme4 package. High five to both!

Final Conclusion: Mixed Model Magic Unveiled

Extracting *BLUEs* and *BLUPs* is just solving the **Mixed Model Equation**—which boils down to some honest, hard-working matrix multiplication. The only sneaky bit? Getting the variance components, which is where fancy software like REML steps in, doing the number crunching for us.

What we've shown is that by building the right matrices and then either solving the equation ourselves or letting packages like lme4 do the job, you end up with the same results.

It's all the same math wrapped in different software—no tricks, no secrets, just good old matrix multiplication.

So relax, enjoy the clarity, and get ready for another adventure soon—because mixed models are classic, simple, and always magical!

Additional Literature on Mixed Models

Here, I am giving link to some of useful resources on Mixed model analysis in Crops

- Application of mixed models in Plant Breeding

- Towards understanding and use of mixed-model analysisof agricultural experiments
- Mixed models with R
- Introduction to linear mixed models
- Fitting Linear Mixed-Effects Models Using lme4
- Long-Term Experiments with cropping systems: Case studies on data analysis
- A brief introduction to mixed effects modelling and multi-model inference in ecology
- Perils and pitfalls of mixed-effects regression models in biology
- Genetic Data Analysis for Plant and Animal Breeding(Book)
- Linear Mixed-Effects Model (Book)

For any suggestions or comments, please feel to reach at waseem.hussain@cgiar.org