Module 3: Optimization of Training Sets with TrainSel Package

Fundamentals of Genomic Prediction and Data-Drive Crop Breeding

(November 24-28, 2025)

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

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

Margaret Catolos
Associate Scientist
International Rice Research Institute
South-Asian Hub, Hyderabad
m.catolos@cgiar.org

November 20, 2025

Contents

Load the Libraries 1

Reading Genotypic Data 1

Extract the Training Population 1

Now we will create a Training set using "TrainSel" R package . . . 2

Optimize the Control Settings using TrainSel . . . 2

How Good Training Set Represents Whole Data 3

Create Biplot . . . 3

Literature 3

Load the Libraries

```
> # Load the Required Libraries
>   rm(list=ls()) # Remove previous work
>   library(dplyr)
>   library(readxl)
>   library(TrainSel)
>   library(factoextra)
>   library(FactoMineR)
>   library(rrBLUP)
```

Reading Genotypic Data

1. Construction of Genomic Relationship Matrix and Imputations for Missing Data

Using the numeric data, we will construct the Genomic Relationship Matrix Function, A.mat, utilized to estimate the genomic relationship matrix (VanRanden, 2008).

2. Imputations for Missing Data

Imputations are done for imputing the missing values, using Expectation-Maximization (EM) Algorithm.

It is an iterative procedure in which it uses other variables to impute a value (Expectation), then checks whether it is the most likely value (Maximization) or not. If not, it re-imputes a more likely value. This goes on until it reaches the most likely value.

EM imputations are better than mean imputations because they preserve the relationship with other variables, vital when we have to use factor analytical or regression models.

Note: EM Imputations still underestimate standard error, however.

Extract the Training Population

1. R package TrainSel : Deniz Akdemir et. al. 2021

TrainSel, provides flexible, efficient, and easy-to-use tools that can be used for the selection of training populations (STP).

TrainSel uses a combination of Genetic Algorithm (GA) (Holland, 1992) and Simulated Annealing (SA) Algorithm (Haines, 1987) for solving combinatorial optimization problems.

Built-in-design Criterion, CDMin (Laloë (1993), Laloë and Phocas (2003), Rincent et al. (2012))

Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8138169/pdf/fgene-12-655287.pdf

***Alternative Packages which can be utilized for Training Set Preparation

a. R Package TSDFGS, Deniz Akdemir et al. (2015), Ou and Liao (2019)

In TSDFGS, we use R function optTrain, for optimal training set determination The function "optTrain" use a genetic algorithm to evaluate an optimal solution using one of the criteria (r-score (Ou J.H., Liao C.T. (2018)), PEV-score (Akdemir D. et al. (2015)), CD-score(Laloe D. (1993))).

Basis: Genetic Algorithm (GA) is an simple exchange algorithm (change an individual every iteration), most suited for combinatorial optimization problem (Akdemir et al. 2015). GA can usually converge to an optimum within a reasonable computing time.

Source: https://cran.r-project.org/web/packages/TSDFGS/TSDFGS.pdf

b. R package STPGA, Deniz Akdemir et. al. 2017

Expands as Selection of Training Populations by Genetic Algorithm. In STPGA we use R function GenAlgForSubset-Selection.

Source: https://cran.r-project.org/web/packages/STPGA/STPGA.pdf

Now we will create a Training set using "TrainSel" R package

Optimize the Control Settings using TrainSel

npop" which is the size of the genetic algorithm population, "nelite" which is the number of elite solutions selected in each iteration, "niterations" which is the maximum number of iterations for the genetic algorithm, "miniterbefstop" is the minimum number of iterations of "no change" before the algorithm is deemed converged, "tolconv" which is the tolerance for determining "no change" in the criteria values, "niterSANN" which is the number of iterations for the SA algorithm, "stepSANN" which controls the speed of cooling of the SA algorithm. For larger problems increasing "niterations" and "niterbefstop" parameters will usually suffice. Optimum value of "niterations" parameter= 5000 and the "minitbefstop" parameter =200 worked optimum.

```
> ############################################
> # Generating a training population
> ############################################
>
> #Objective: Training set optimization consists of choosing a set of training individuals that will be
>
>   library(TrainSel)
>   control=TrainSelControl()
>
> # When using a mixed model based CDmin statistic we need to prepare the data using the 'MakeTrainSel-
> # We will use marker data to prepare data for mixed model based criterion.
>   TSData<-MakeTrainSelData(M=geno)
> ## We use 'TrainSel' function to select an Unordered Sample ("UOS") of size 20 from the first 100 ind
> #as a default it picks, corresponding to the first 100 rows of data. The default settings do not need
>   controlOuter<-TrainSelControl()
>   controlOuter$niterations<-5
>   T.set<-TrainSel(Data=TSData, Candidates=list(1:100), setsizes=c(20),settypes="UOS", control=
+                  control)
>
> # Subset the best complete genotypic data of the best picked 10 genotypes from the genotype file.
>   T.set<- T.set$BestSol_int
> # Subset the best complete genotypic data of the best picked 100 genotypes from the genotype file
>   TS.final <- geno[T.set,]
```

How Good Training Set Represents Whole Data

- Success of GS depends upon the relationship of Training set with Testing Set!
- We will visualize the relationship using the biplot obtained from **Genomic relationship matrix**!
- **Genomic Relationship Matrix (GRM)** based on **VanRanden (2008)** will be created!
- The Steps used to create this GRM is: 1. Create a center of marker data 2. Create a Cross Product (XX) annd 3. Divide the (XX) by number of markers !

$$GRM = XX^t/m$$

- More on relationship matrix can be found here Source 1, Source2 !

```
> # First Get genomic matrix
> #Xs <- scale(geno.oyt, center = TRUE)
> # Construct G matrix
>   GM <- geno %*% t(geno)/ncol(geno)
>   dim(GM)
```

Create Biplot

```
> #  Get lines from training set
>   line.names<-data.frame(Genotype=ifelse(row.names(geno)%in%row.names(TS.final), "Un-selected", "Sele
> # Perfom PCA on GM matrix
>   pca<- PCA(GM, graph = FALSE)
> # Get the Biplot
>   biplot.gm<-fviz_pca_biplot(pca, palette = "jco", axes = c(1, 2), geom="point",pointsize = 3,
+                 addEllipses = FALSE,col.ind = line.names$Genotype,
+                 geom.var= "none", label = "none", legend.title = "Group")+
+   theme_minimal()+
+   # add and modify the title to plot
+   theme (
+     plot.title = element_blank(),
+     # add and modify title to x axis
+     axis.title.x = element_text(color="black", size=12),
+     # add and modify title to y axis
+     axis.title.y = element_text(color="black", size=12)) +
+   # modify the axis text
+   theme(axis.text= element_text(color = "black", size = 12))
>   biplot.gm
```

Literature

1. https://github.com/TheRocinante-lab/TrainSel/blob/main/TrainSelUsage.pdf

---

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