TrainSel Usage

Deniz Akdemir

4/11/2021

Introduction

In this section, we will illustrate the use of the package 'TrainSel'. We will use the data sets provided within the package under the object named 'WheatData' throughout this presentation. The original data was obtained from the webpage https://triticeaetoolbox.org/. The data contains the genomewide marker data (at 4670 markers) and a simulated trait data (phenotypic measurements are simulated using an infinitismall model assuming an heritability value of 0.7.) for 200 wheat varieties.

We can load the library and 'WheatData' using the following code:

```
library(TrainSel)
```

```
## Loading required package: cluster
```

```
data("WheatData")
```

Marker data is contained in the matrix object 'Wheat.M', a relationship matrix for the genotypes calculated from the marker matrix is in 'Wheat.K', and the plant height measurements are in 'Wheat.Y'. We can see the format of these data:

```
Wheat.M[1:5,1:5]
```

```
##            IWA1 IWA2 IWA3 IWA4 IWA5
## IWA8610266    1    1   -1   -1   -1
## IWA8606816    1    1    1   -1   -1
## 3883          1    1    1   -1   -1
## NW86A         1    1    1    1   -1
## PI345476      1    1    1   -1   -1
```

```
Wheat.K[1:5,1:5]
```

```
##            IWA8610266 IWA8606816       3883      NW86A   PI345476
## IWA8610266  1.9578361  0.8261588  0.7546713  0.5149389 -0.2697654
## IWA8606816  0.8261588  2.0033376  0.5756170  0.5241991 -0.2315324
## 3883        0.7546713  0.5756170  1.9807687  0.7077443 -0.2888690
## NW86A       0.5149389  0.5241991  0.7077443  2.2816612 -0.2645939
## PI345476   -0.2697654 -0.2315324 -0.2888690 -0.2645939  1.8086996
```

```
Wheat.Y[1:5,]
```

```
##                id plant.height
## 1846 10542-63/87        91.44
## 250          131A       124.46
## 1541         1548       129.54
## 1516       155/71       124.46
## 508          1664       134.62
```

Selection of a subset of genotypes for a phenotypic experiment in a single environment

Our aim is to build a predictive model for the plant height based on the genomewide marker data. The dataset contains the plant heights for all of these genotypes, however, for a moment, assume that we do not have the plant heights measurements for these 200 genotypes and currently only a subset of 160 candidate genotypes are available to be used in the phenotypic experiment. Furthermore, since measuring plant height for 160 candidate genotypes via a phenotypic experiment can be costly, we assume that we can only perform the phenotypic experiment with say a maximum of 50 training genotypes selected from the 160. Following this phenotypic experiment, we can use the available marker data and the measured height of these 50 genotypes to train a genomic prediction model to make inferences about the heights of the remaining 110 genotypes or any other set of genotypes that we have the same genomewide marker data, for example, the 40 genotypes that were not available for the phenotypic experiment.

Selection of a subset of genotypes for a homogeneous design in a single environment

The simplest use case for 'TrainSel' is the situation where phenotypic experiment will only performed in one homogeneous environment. In this case, our purpose is to select a subset of size 50 from the 160 available genotypes so that the genomic prediction models that are trained on this 50 has a good generalization performance.

We distinguish between two cases of optimal training set selection based on whether we seek that generalize well for a specific target set of genotypes (Targeted optimization) or not (Un-targeted optimization). Not all optimization criteria are sensitive to this distinction, however when it is so this is reflected in how the optimization criteria is calculated.

Un-targeted optimization There are many different statistics that can be used for the untargeted optimization with a homogeneous design in a single environment. The package 'TrainSel' is designed to be flexible to be used with any design criteria, however, this means that the users need to program their own optimization functions. Below are some example functions that should get started for writing your own:

D-optimality criterion D-optimality criterion is a model based design criterion. The underlying model for the D-optimality criterion is a linear model. For the problem of selection of a subset of genotypes for a homogeneous design in a single environment a linear model relating the genotypic data to phenotypic measurements in the training data can be expressed as

$$y = 1\mu + f(M)\beta + \epsilon,$$

where y is the n vector of phenotypic measurements in the training data, μ is a scalar parameter for the mean of the phenotypic measurements, M is the $n \times m$ the marker matrix for the n training genotypes, $f(M)$ is a genomic features matrix with dimensions $n \times q$, β is the q vector of effects of genomic features, and ϵ is the residual error vector of length n. We further assume that the elements of ϵ are independent and identically distributed with a normal distribution with zero mean and variance σ_e^2. Under this model the variance of the estimators for β is known to be proportional to $[f(M)'f(M)]^{-1}$. D-optimal selection of n training genotypes from N individuals in the candidate set involves minimizing the determinant of this matrix (or equivalently maximizing the log-determinant of $f(M)'f(M)$). The feature matrix $f(M)$ is usually the first q principal components matrix for the marker matrix M and for this measure to be defined q should be less than n.

```
#We will use the first 30 principal components for this

Wheat.M_centered<-scale(Wheat.M, center=TRUE, scale=FALSE)
svdWheat.M_centered<-svd(Wheat.M_centered, nu=5, nv=5)

PC<-Wheat.M_centered%*%svdWheat.M_centered$v
dim(PC)
```

```
## [1] 200   5
```

```
dataDopt<-list(FeatureMat=PC[1:160,]) # candidates are in rows 1:160

DOPT<-function(soln, Data){
  Fmat<-Data[["FeatureMat"]]
  return(determinant(crossprod(Fmat[soln,]), logarithm=TRUE)$modulus)
}


library(TrainSel)

TSC<-TrainSelControl()
TSC$niterations=3
TSC$npop=10
TSC$nelite=3

TSOUTD<-TrainSel(Data=dataDopt,
           Candidates = list(1:160),
           setsizes = c(50),
           settypes = "UOS",
           Stat = DOPT, control=TSC)
```

```
## Maximum number of iterations reached.
```

```
head(rownames(Wheat.M)[TSOUTD$BestSol_int])
```

```
## [1] "3883"         "1666"         "IWA8610164"   "F27"          "IWA8604778"
## [6] "POLESSKAJA71"
```

CDMEAN-optimality criterion CDMEAN-optimality criterion is also a model based criterion it is based on a G-BLUP mixed model. For the problem of selection of a subset of genotypes for a homogeneous design in a single environment, a G-BLUP model relating the genotypic data to phenotypic measurements in the training data can be expressed as

$$y = 1\mu + Zu + \epsilon$$

with μ a scalar parameter for the mean of the phenotypic measurements, Z the $n \times N$ design matrix for the N genotypes in the candidate set, $\epsilon \sim N_n(0, \sigma_e^2)$ independent of $u \sim N_q(0; \sigma_g^2 G)$.

For this model, the coefficient of determination matrix of $\widehat{u}$ for predicting u is given by

$$(GZ'PZG) \oslash G$$

where $P = V^{-1} - V^{-1}1(1'V^{-1}1)^{-1}1'V^{-1}$ is the projection matrix and $\oslash$ expresses the element-wise division.

The diagonals of this matrix are the coefficient of determination of the predictions for individual genotypes and the mean of these coefficient of determination values over the selected genotypes is called the CDMEAN-optimality criterion[1]. CDMEAN criterion takes values between 0 and 1 and the larger values are preferable. For the un-targeted optimization, the usual practice is to use CDMEAN that is calculated over the genotypes not included in the training set. We can program this objective function to be used in 'TrainSel' as follows:

```
dataCDMEANopt<-list(G=Wheat.K[1:160,1:160], lambda=1)# 1:160 are the Candidates

CDMEANOPT<-function(soln, Data){
  G<-Data[["G"]]
  lambda<-Data[["lambda"]]
```

[1] A risk averse approach would entail maximizing the minimum of selected diagonals.

```
  Vinv<-solve(G[soln,soln]+lambda*diag(length(soln)))
  outmat<-(G[,soln]%*%(Vinv-(Vinv%*%Vinv)/sum(Vinv))%*%G[soln,])/G
  return(mean(diag(outmat[-soln,-soln])))
}



TSOUTCD<-TrainSel(Data=dataCDMEANopt,
           Candidates = list(1:160),
           setsizes = c(50),
           settypes = "UOS",
           Stat = CDMEANOPT, control=TSC)
```

```
## Maximum number of iterations reached.
```

```
head(rownames(Wheat.M)[TSOUTCD$BestSol_int])
```

```
## [1] "3883"       "1666"       "69Z6.441"   "F27"        "B-871"
## [6] "RCAT000491"
```

Maximin distance criterion Maximin distance criterion is a non-parametric design criteria. An optimal training set of size n from the N candidates is selected by maximizing the minimum [2] genetic distance among the training genotypes, so this is a space filling design. Next, we show how to program this criterion in R:

```
dataMaximin<-list(DistMat=as.matrix(dist(Wheat.M_centered[1:160,])))

MaximinOPT<-function(soln, Data){
  Dsoln<-Data[["DistMat"]][soln,soln]
  DsolnVec<-Dsoln[lower.tri(Dsoln,diag=FALSE)]
  return(min(DsolnVec))
}



TSOUTMaximin<-TrainSel(Data=dataMaximin,
           Candidates = list(1:160),
           setsizes = c(50),
           settypes = "UOS",
           Stat = MaximinOPT, control=TSC)
```

```
## Maximum number of iterations reached.
```

```
head(rownames(Wheat.M)[TSOUTMaximin$BestSol_int])
```

```
## [1] "NW86A"    "1666"     "NS18-99"  "B-871"    "VIRGO"    "PI254048"
```

Targeted optimization When the focus is on making inferences about the trait values for a known target set of genotypes is using genomic prediction, we can use what we call a targeted optimization criteria.

Mean PEV criterion based on linear model Dopt criterion is not sensitive to information about the target set of genotypes. Nevertheless, a related linear model based criteria called the mean prediction error variance (Mean PEV) can be used when the genotypic data for the target set is available. This criteria relates to the average prediction error variance of the predictions for the target set of genotypes using the linear model in Equation 1.

[2] We could also maximize the mean distance leading to Maximean criterion

```
dataPEVlm<-list(FeatureMat=PC, Target=161:200)  #PC has 200 rows




PEVlmOPT<-function(soln, Data){
  Fmat<-Data[["FeatureMat"]]
  targ<-Data[["Target"]]
  return(mean(diag(Fmat[targ,]%*%solve(crossprod(Fmat[soln,]))%*%t(Fmat[targ,]))))
}


TSOUTPEVlm<-TrainSel(Data=dataPEVlm,
           Candidates = list(1:160),
           setsizes = c(50),
           settypes = "UOS",
           Stat = PEVlmOPT, control=TSC)
```

```
## Maximum number of iterations reached.
```

```
head(rownames(Wheat.M)[TSOUTPEVlm$BestSol_int])
```

```
## [1] "3883"      "PI345476"  "H86-708"   "237-VII/2" "204/71"    "VIRGO"
```

```
dataCDMEANTargetOpt<-list(G=Wheat.K, lambda=1, Target=161:200)

CDMEANOPTTarget<-function(soln, Data){
  G<-Data[["G"]]
  lambda<-Data[["lambda"]]
  targ<-Data[["Target"]]
  Vinv<-solve(G[soln,soln]+lambda*diag(length(soln)))
  outmat<-(G[,soln]%*%(Vinv-(Vinv%*%Vinv)/sum(Vinv))%*%G[soln,])/G
  return(mean(diag(outmat[targ,targ])))
}



TSOUTCDTarg<-TrainSel(Data=dataCDMEANTargetOpt,
           Candidates = list(1:160),
           setsizes = c(50),
           settypes = "UOS",
           Stat = CDMEANOPTTarget, control=TSC)
```

Targeted CDMEAN criterion

```
## Maximum number of iterations reached.
```

```
head(rownames(Wheat.M)[TSOUTCDTarg$BestSol_int])
```

```
## [1] "3883"       "PI345476"   "237-VII/2"  "1666"       "69Z6.441"
## [6] "IWA8604778"
```

Selection of a subset of genotypes for an known design in a single environment

In certain cases, we are looking for conducting a phenotypic experiment with n training genotypes selected out of N candidate genotypes but in addition, we also have a particular blocking structure and environmental covariates involved in the design of the experiment. Suppose the matrix E is the $n \times p$ environmental covariates matrix. For instance, this matrix could be the design matrix for a row-column blocking within the environment. We assume still that we want make inferences about the genomic values after accounting for these covariates. In this case, the order in which the genotpes are positioned in the environment will be important. Perhaps, we would like to use similar genotypes in genotypes in dissimilar blocks and also we would like to observe as genetically distant genotypes within similar blocks.

Un-targeted optimization

D-optimality criterion with environmental covariates D-optimality criterion can be easily adopted for this purpose. FIrts we write the model as

$$y = E\beta_{env} + f(M)\beta_f + \epsilon$$

where y is the n vector of phenotypic measurements in the training data, E is the $n \times p$ design matrix for the environmental covariates, β_{env} is the p vector of the effects of the environmental covariates, M is the $n \times m$ the marker matrix for the n training genotypes, $f(M)$ is a genomic features matrix with dimensions $n \times q$, β_f is the q vector of effects of genomic features, and ϵ is the residual error vector of length n. We further assume that the elements of ϵ are independent and identically distributed with a normal distribution with zero mean and variance σ^2_ϵ. Under this model the variance of the estimators for β is known to be proportional to $[f(M)'(I - E(E'E)^{-1}E')f(M)]^{-1}$. D-optimal selection of n training genotypes from N individuals in the candidate set involves minimizing the determinant of this matrix (or equivalently maximizing the log-determinant of $f(M)'(I - E(E'E)^{-1}E')f(M)$). The matrix $(I - E(E'E)^{-1}E')$ is the projection matrix to the orthogonal space of column space of E.

```
E<-data.frame(expand.grid(row=paste("row",1:5, sep="_"),
                          col=paste("col",1:10, sep="_")))
E$row<-as.factor(E$row)
E$col<-as.factor(E$col)


DesignE<-model.matrix(~row+col+row*col, data=E)

P<-diag(nrow(DesignE))-DesignE%*%solve(crossprod(DesignE))%*%t(DesignE)

dataDoptEnv<-list(FeatureMat=PC[1:160,], Projection=P)##1:160 in candidate

DOPTwithE<-function(soln, Data){
  Fmat<-Data[["FeatureMat"]]
  P<-Data[["Projection"]]
  return(determinant(crossprod(P%*%Fmat[soln,]), logarithm=TRUE)$modulus)
}

TSOUTDwithE<-TrainSel(Data=dataDoptEnv,
           Candidates = list(1:160),
           setsizes = c(50),
           settypes = "OS",
           Stat = DOPTwithE, control=TSC)
```

```
## Maximum number of iterations reached.
```

```
TSOUTDwithE$BestSol_int #order of this is important
```

```
##  [1]  46  72 149  43 119  47  26 100  79 106 112 150 157  45   3 116 160 133  22
## [20] 103  94  48 107  28  55  52  39 101  21  12   8 122 132   9  61 138  56  20
## [39]  80 148 104 115  78 105  10  65  16 146 159 140
```

```
head(rownames(Wheat.M)[TSOUTDwithE$BestSol_int])
```

```
## [1] "IWA8604850"         "IWA8610471"         "I/6"
## [4] "SQUAREHEADS_MASTER" "VAKKA"              "69Z2.88/7B"
```

```
E$GID<-rownames(Wheat.M)[TSOUTDwithE$BestSol_int]
###Here is the final design
head(E)
```

```
##     row   col                GID
## 1 row_1 col_1         IWA8604850
## 2 row_2 col_1         IWA8610471
## 3 row_3 col_1                I/6
## 4 row_4 col_1 SQUAREHEADS_MASTER
## 5 row_5 col_1              VAKKA
## 6 row_1 col_2         69Z2.88/7B
```

CDMEAN-optimality criterion with environmental covariates We can also use environmental covariates with the CDMEAN-optimality criterion. In ordor to do this we first need to add the environmental covariates into the G-BLUP model. This model is written as

$$y = E\beta_{env} + +Zu + \epsilon$$

with E is the $n \times p$ design matrix for the environmental covariates, β_{env} is the p vector of the effects of the environmental covariates, Z is the $n \times N$ design matrix for the N genotypes in the candidate set, $\epsilon \sim N_n(0, \sigma_e^2)$ is independent of $u \sim N_q(0; \sigma_g^2 G)$.

For this model, the coefficient of determination matrix of $\widehat{u}$ for predicting u is given by

$$(GZ'PZG) \oslash G$$

where $P = V^{-1} - V^{-1}E(E'V^{-1}E)^{-1}E'V^{-1}$ is the projection matrix and $\oslash$ expresses the element-wise division.

```
dataCDMEANoptwithEnv<-list(G=Wheat.K[1:160,1:160],E=DesignE, lambda=1)

CDMEANOPTwithEnv<-function(soln, Data){
  G<-Data[["G"]]
  E<-Data[["E"]]

  lambda<-Data[["lambda"]]
  Vinv<-solve(G[soln,soln]+lambda*diag(length(soln)))
  outmat<-(G[,soln]%*%
             (Vinv-Vinv%*%E%*%solve(t(E)%*%Vinv%*%E)%*%t(E)
              %*%Vinv)%*%G[soln,])/G
  return(mean(diag(outmat[-soln,-soln])))
}



TSOUTCDwithENV<-TrainSel(Data=dataCDMEANoptwithEnv,
```

```
           Candidates = list(1:160),
           setsizes = c(50),
           settypes = "OS",
           Stat = CDMEANOPTwithEnv, control=TSC)
```

```
## Maximum number of iterations reached.
```

```
E$GID<-rownames(Wheat.M)[TSOUTCDwithENV$BestSol_int]
###Here is the final design
head(E)
```

```
##     row   col        GID
## 1 row_1 col_1 IWA8600078
## 2 row_2 col_1     TN1649
## 3 row_3 col_1     HECTOR
## 4 row_4 col_1      VAKKA
## 5 row_5 col_1     PRIMUS
## 6 row_1 col_2   LOVRIN13
```

Targeted optimization and allowing for replicates We can also do targeted optimization in this case with minimal change to the last code:

```
dataCDMEANoptwithEnvTarget<-list(G=Wheat.K,E=DesignE,Target=161:200, lambda=1)

CDMEANOPTwithEnvTarget<-function(soln, Data){
  G<-Data[["G"]]
  E<-Data[["E"]]
  targ<-Data[["Target"]]
  lambda<-Data[["lambda"]]
  Vinv<-solve(G[soln,soln]+lambda*diag(length(soln)))
  outmat<-(G[,soln]%*%
             (Vinv-Vinv%*%E%*%solve(t(E)%*%Vinv%*%E)%*%t(E)%*%Vinv)
           %*%G[soln,])/G
  return(mean(diag(outmat[targ,targ])))
}



TSOUTCDwithENVTarg<-TrainSel(Data=dataCDMEANoptwithEnvTarget,
           Candidates = list(1:160),
           setsizes = c(50),
           settypes = "OMS",
           Stat = CDMEANOPTwithEnvTarget, control=TSC)
```

```
## Maximum number of iterations reached.
```

```
E$GID<-rownames(Wheat.M)[TSOUTCDwithENVTarg$BestSol_int]
###Here is the final design
head(E)
```

```
##     row   col          GID
## 1 row_1 col_1   IWA8611008
## 2 row_2 col_1 KOOPERATORKA
## 3 row_3 col_1         3883
## 4 row_4 col_1     PI254049
## 5 row_5 col_1      STARING
```

```
## 6 row_1 col_2   IWA8613659
```

Design for a phenotypic experiment in multiple environments

In practice, most genomic selection experiments are performed over multiple environments. Designing genomic selection over multiple environments means we would like to choose training genotypes to use in each of these environments and for genomic selection the distribution of the alleles within and between the different environments can be arranged optimally for obtaining better generalization performance.

Using CDMEAN-optimality criterion for genomic selection experiment design in multiple environments As before, we first need to state the underlying model. For multi-environmental trials, a commonly used genomic prediction model is the multi-environmental G-BLUP model. Suppose we have 3 environments, in Environment 1 we can accommodate 30 genotypes in a 6-rows 5-columns design, in Environment 2 we can accommodate 20 genotypes in a 4-rows 5-columns design, and in Environment 3 we can accommodate 50 genotypes in an unknown homogeneous design. We assume that the genomic covariance of the environments is (proportionally) equal to the matrix

$$V_g = \begin{pmatrix} 1.0 & .7 & .5 \\ .7 & 1.2 & .8 \\ .5 & .8 & 1.5 \end{pmatrix}$$

We assume that the residual errors in these environments are correlated with the following covariance matrix

$$V_e = \begin{pmatrix} 1.0 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1.0 \end{pmatrix}.$$

We can express the model for the training data as follows:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = E\beta_{env} + \begin{pmatrix} Z_1 u_1 \\ Z_2 u_2 \\ Z_3 u_3 \end{pmatrix} + \begin{pmatrix} Z_1 \epsilon_1 \\ Z_2 \epsilon_2 \\ Z_3 \epsilon_3 \end{pmatrix}$$

where E is the $n \times p$ design matrix for the environmental covariates, β_{env} is the p vector of the effects of the environmental covariates, y_i is an n_i vector and Z_i is the $n_i \times N$ design matrix for the N genotypes in the candidate set in environment i for $i = 1, 2, 3$. In addition, we assume $(\epsilon_1, \epsilon_2, \epsilon_3) \sim N_{N\times 3}(0, I_N, V_e)$ is independent of $(u_1, u_2, u_3) \sim N_{N\times 3}(0; G, V_g)$. This means that the vectorized form of these matrices $\epsilon = vec(\epsilon_1, \epsilon_2, \epsilon_3)$ and $u = vec(u_1, u_2, u_3)$ are independently distributed as $N_{3N}(0, V_e \otimes I_N)$ and $N_{3N}(0, V_g \otimes G)$.

In our case $n_1 = 30, n_2 = 20, n_3 = 50$, and $n = n_1 + n_2 + n_3 = 100$. Here is how you can approach this problem using 'TrainSel':

```
Vg=matrix(c(1.0 , .7 , .5 , .7 , 1.2 ,.8 , .5 , .8 , 1.5), 3,3)
Ve=matrix(c(1.0, 0,0  , 0,1.5,0 , 0,0,1.0), 3,3)
rownames(Vg)<-colnames(Vg)<-rownames(Ve)<-colnames(Ve)<-paste("E",1:3, sep="")
G<-kronecker(Vg, Wheat.K, make.dimnames = TRUE)
R<-kronecker(Ve, diag(nrow(Wheat.K)), make.dimnames = TRUE)
#### Note the shape of G (same as R)

head(rownames(G))
```

```
## [1] "E1:IWA8610266" "E1:IWA8606816" "E1:3883"       "E1:NW86A"
## [5] "E1:PI345476"   "E1:H86-708"
```

```
tail(rownames(G))
```

```
## [1] "E3:CAR1012"           "E3:416-V/69"          "E3:IWA8613952"
## [4] "E3:HODOMINSKA_HOLICA" "E3:204-VI/9-A"        "E3:PI254045"
```

```
dim(G)
```

```
## [1] 600 600
```

By examining the shape of the G matrix above we see that the candidate set are on the 1st through 160th, 200th through 360th and,400th through 560th rows and columns of this matrix. We want to use 30 from 1 through 160, 20 from 200 through 360 and 50 from 400 through 560. The first two sets are ordered and last one is unordered. We are also going to assume duplicates within an environment are not allowed.

Design Matrix for the environments is obtained below. I am assuming no interaction effects between rows and columns.

```
E1<-(expand.grid(row=paste("row",1:6, sep="_"),
                 col=paste("col",1:5, sep="_")))


E2<-(expand.grid(row=paste("row",1:4, sep="_"),
                 col=paste("col",1:5, sep="_")))


E3<-data.frame(row=paste("row",rep(1,50),sep="_"),
               col=paste("col",rep(1,50), sep="_"))



EnvData<-data.frame(Env=c(rep("E1", 30), rep("E2", 20),rep("E3", 50)),
                 rbind(E1,E2,E3))
DesignE<-model.matrix(~Env+Env/(row+col), data=EnvData)
DesignE<-DesignE[,colSums(DesignE)>0]
colnames(DesignE)
```

```
##  [1] "(Intercept)"    "EnvE2"          "EnvE3"          "EnvE1:rowrow_2"
##  [5] "EnvE2:rowrow_2" "EnvE1:rowrow_3" "EnvE2:rowrow_3" "EnvE1:rowrow_4"
##  [9] "EnvE2:rowrow_4" "EnvE1:rowrow_5" "EnvE1:rowrow_6" "EnvE1:colcol_2"
## [13] "EnvE2:colcol_2" "EnvE1:colcol_3" "EnvE2:colcol_3" "EnvE1:colcol_4"
## [17] "EnvE2:colcol_4" "EnvE1:colcol_5" "EnvE2:colcol_5"
```

```
dataCDMEANoptwithEnvME<-list(G=G,R=R, E=DesignE)

CDMEANOPTwithEnvME<-function(soln, Data){
  G<-Data[["G"]]
  R<-Data[["R"]]
  E<-Data[["E"]]
  Vinv<-solve(G[soln,soln]+R[soln,soln])
  outmat<-(G[,soln]
           %*%(Vinv-Vinv%*%E%*%solve(t(E)%*%Vinv%*%E)%*%t(E)%*%Vinv)
           %*%G[soln,])/G
  return(mean(diag(outmat[-soln,-soln])))
}



TSOUTCDwithENVME<-TrainSel(Data=dataCDMEANoptwithEnvME,
           Candidates = list(1:160, 201:360,401:560),
           setsizes = c(30,20,50),
           settypes = c("OS","OS","UOS"),
```

```
           Stat = CDMEANOPTwithEnvME, control=TSC)
```

```
## Maximum number of iterations reached.
```

```
E1$GID<-rownames(G)[TSOUTCDwithENVME$BestSol_int[1:30]]
###Here is the final design for env 1
head(E1)
```

```
##     row   col                      GID
## 1 row_1 col_1           E1:IWA8606752
## 2 row_2 col_1          E1:69Z2.102/85A
## 3 row_3 col_1 E1:KRASNOVODOPADSKAJA210
## 4 row_4 col_1            E1:ZG2834/74
## 5 row_5 col_1             E1:CHAMBORD
## 6 row_6 col_1           E1:IWA8604770
```

```
E2$GID<-rownames(G)[TSOUTCDwithENVME$BestSol_int[31:50]]
###Here is the final design for env 2
head(E2)
```

```
##     row   col               GID
## 1 row_1 col_1   E2:KOOPERATORKA
## 2 row_2 col_1        E2:68F6691
## 3 row_3 col_1         E2:CAR715
## 4 row_4 col_1           E2:1666
## 5 row_1 col_2           E2:3883
## 6 row_2 col_2 E2:VARMA|PI265484
```

```
E3$GID<-rownames(G)[TSOUTCDwithENVME$BestSol_int[51:100]]
###Here is the final design for env 3
head(E3)
```

```
##     row   col           GID
## 1 row_1 col_1 E3:IWA8606816
## 2 row_1 col_1      E3:NW86A
## 3 row_1 col_1 E3:IWA8610164
## 4 row_1 col_1 E3:IWA8604778
## 5 row_1 col_1     E3:204/71
## 6 row_1 col_1 E3:KM618-2-90
```

```
#plot(TSOUTCDwithENVME$maxvec, ylab="CD")
```

Implementing a penalty function for the total number of genotypes in the experiment

```
PenaltyFunction<-function(soln){
  soln1<-soln[1:30]
  soln2<-soln[31:50]-200
  soln3<-soln[51:100]-400
  numuniquegeno<-length(unique(c(soln1,soln2,soln3)))
  if( (numuniquegeno<80) |  (90<numuniquegeno)){
    return(min(numuniquegeno-80, 90-numuniquegeno))
     } else {
    return(0)
    }

}
```

```
CDMEANOPTwithEnvMEwithPenalty<-function(soln, Data){
  penalty<-PenaltyFunction(soln)
  if (penalty==0){
  G<-Data[["G"]]
  R<-Data[["R"]]
  E<-Data[["E"]]
  Vinv<-solve(G[soln,soln]+R[soln,soln])
  outmat<-(G[,soln]
           %*%(Vinv-Vinv%*%E%*%solve(t(E)%*%Vinv%*%E)%*%t(E)%*%Vinv)
           %*%G[soln,])/G
  return(mean(diag(outmat[-soln,-soln])))
  } else {return(penalty)}
}



TSOUTCDwithENVMEwithPenalty<-TrainSel(Data=dataCDMEANoptwithEnvME,
           Candidates = list(1:160, 201:360,401:560),
           setsizes = c(30,20,50),
           settypes = c("OS","OS","UOS"),
           Stat = CDMEANOPTwithEnvMEwithPenalty, control=TSC)
```

```
## Maximum number of iterations reached.
```

```
E1$GID<-rownames(G)[TSOUTCDwithENVMEwithPenalty$BestSol_int[1:30]]
###Here is the final design for env 1
head(E1)
```

```
##     row   col               GID
## 1 row_1 col_1         E1:CAR105
## 2 row_2 col_1 E1:WHITE_VICTORIA
## 3 row_3 col_1 E1:69Z2.179/1380C
## 4 row_4 col_1     E1:IWA8607148
## 5 row_5 col_1       E1:KM708-90
## 6 row_6 col_1   E1:MACEDONIAN14
```

```
E2$GID<-rownames(G)[TSOUTCDwithENVMEwithPenalty$BestSol_int[31:50]]
###Here is the final design for env 2
head(E2)
```

```
##     row   col             GID
## 1 row_1 col_1         E2:1744
## 2 row_2 col_1    E2:JUBILE_II
## 3 row_3 col_1       E2:TN1649
## 4 row_4 col_1          E2:F27
## 5 row_1 col_2 E2:ZERNOGRADKA8
## 6 row_2 col_2       E2:CAR411
```

```
E3$GID<-rownames(G)[TSOUTCDwithENVMEwithPenalty$BestSol_int[51:100]]
###Here is the final design for env 3
head(E3)
```

```
##     row   col           GID
## 1 row_1 col_1 E3:IWA8606816
## 2 row_1 col_1 E3:IWA8604778
## 3 row_1 col_1      E3:LOGAN
```

```
## 4 row_1 col_1 E3:KM618-2-90
## 5 row_1 col_1 E3:IWA8607148
## 6 row_1 col_1   E3:PI254049
```

```
#plot(TSOUTCDwithENVMEwithPenalty$maxvec, ylab="CD")
```

Other usage examples in plant breeding

'TrainSel' can be adopted to be used in optimization of problems other than the design of training populations for genomic prediction. The following examples demonstrate use of the package in some mixed integer optimization problems related to plant breeding.

Unsupervised Variable Selection

In some cases, we are interested in a subset of markers that represent the information contained in the all of the available markers. We can use the distance correlation measure between the marker data with all of the markers and the marker data with a selected set of markers to aid us in this process.We want to maximize the distance correlation measure for a given set of markers of size say 100. You can do this with the following code:

```
library(energy)
dataVarSel<-list(dist(Wheat.M_centered), Wheat.M_centered)
funVarSel<-function(soln,Data){
out<-bcdcor(Data[[1]],dist(Data[[2]][,soln]))
return(out)
}

TSC$npop<-100
TSC$nelite<-5
TSC$niterations<-10
TSOUTVarSel<-TrainSel(Data=dataVarSel,
           Candidates = list(1:ncol(Wheat.M_centered)),
           setsizes = c(100),
           settypes = c("UOS"),
           Stat = funVarSel, control=TSC)
```

```
## Maximum number of iterations reached.
```

```
#to check convergence
#plot(TSOUTVarSel$maxvec, "Distance Correlation")
AnchorMarkers<-TSOUTVarSel$BestSol_int
```

Optimal parental proportions and the Pareto front

In this example we want to select 10 parents out of the 40 genotypes in the target set and provide parental proportions for those 10 parents so that the mean GEBVs (estimated using a model based on the anchor markers) for the progeny of these parents mixed in these proportions would be maximized and inbreeding of the progeny is minimized.

We first need to estimate the marker effects. In this example we will only use the anchor markers, and the model is trained on all 160 candidate genotypes.

```
library(EMMREML)
```

```
## Loading required package: Matrix
```

```
markereffects<-rnorm(100)
Wheat.Y[,2]<-Wheat.M_centered[,AnchorMarkers]%*%markereffects
```

```
Wheat.Y[,2]<-Wheat.Y[,2]+10+rnorm(200, mean=0, sd=sd(Wheat.Y[,2]))
mixedmodelout<-emmreml(y=Wheat.Y[1:160,2],
                       X=matrix(1, nrow=160, ncol=1),
                       Z=Wheat.M_centered[1:160,AnchorMarkers],
                       K=diag(length(AnchorMarkers)))
markereffects<-mixedmodelout$uhat
```

Using the estimated marker effects we can calculate GEBVs for the 40 prospective parents in the Target set. We also subset the genomic relationship matrix for these 40 genotypes from the whole genomic relationship matrix.

```
GEBVs40<-Wheat.M_centered[161:200,AnchorMarkers]%*%markereffects
K40<-Wheat.K[161:200,161:200]
```

Since we know the phenotypic values for the Target in our case, we can povide an estimate of the accuracy for the GEBVs in the Target set:

```
cor(GEBVs40,Wheat.Y[161:200,2])
```

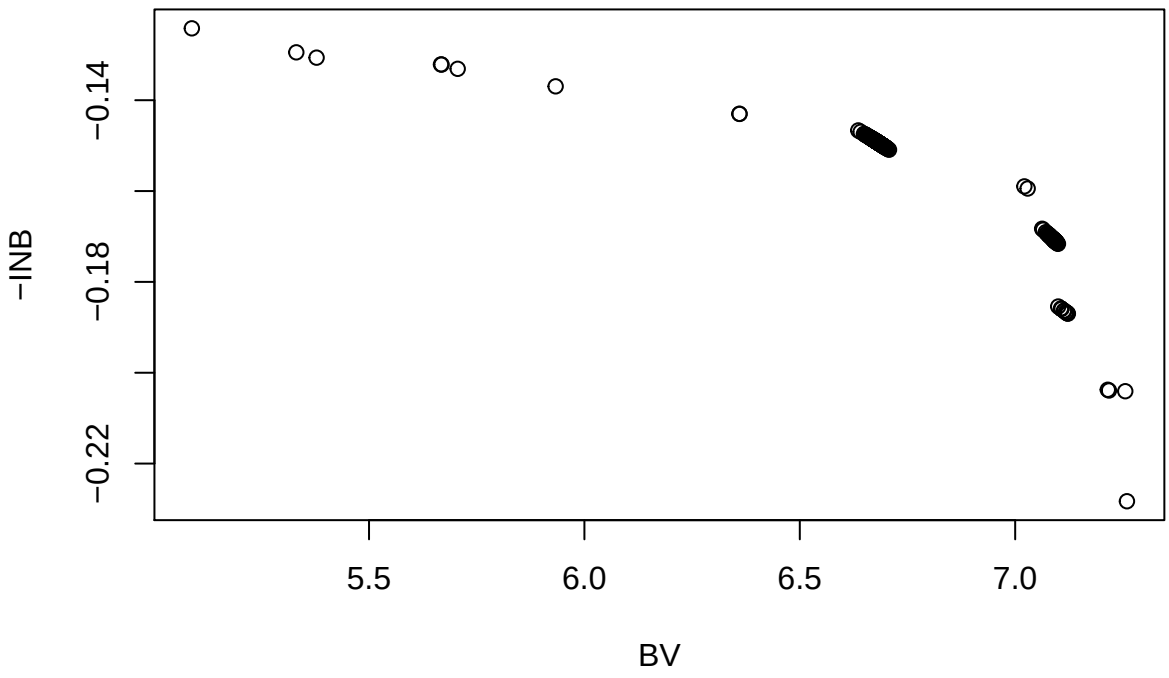
```
##           [,1]
## [1,] 0.773396
```

Now, we are ready to setup the problem:

```
dataOptProp<-list(GEBVs40,K40)
funOptProp<-function(soln_int,soln_dbl,Data){
  props<-soln_dbl/sum(soln_dbl) #to rescale the solution to sum up to 1.
  BV<-crossprod(Data[[1]][soln_int],props)
  Inb<-t(props)%*%Data[[2]][soln_int,soln_int]%*%props
  return(c(BV,-c(Inb)))
}


TSOUTOptProp<-TrainSel(Data=dataOptProp,
           Candidates = list(1:40, c(.05,.9)),
           setsizes = c(10, 10),
           settypes = c("OS","DBL"),
           Stat = funOptProp, nStat = 2)
```

```
## Maximum number of iterations reached.
```

```
plot(t(TSOUTOptProp$BestVal), xlab="BV", ylab="-INB")
```



Optimal genomic mating

```
library(rrBLUP)
##All possible mates

#males 161:200
#females 161:200

#this gives 1600 possible mates.

DFMates<-expand.grid(rownames(Wheat.M[161:200,]),rownames(Wheat.M[161:200,]))


M1<-Wheat.M[161:200,AnchorMarkers]
M2<-Wheat.M[161:200,AnchorMarkers]
#M1[sample(M1%in%c(-1,1),2000)]<-0
#M2[sample(M2%in%c(-1,1),2000)]<-0
M<-rbind(M1, M2)
K<-A.mat(M)
```

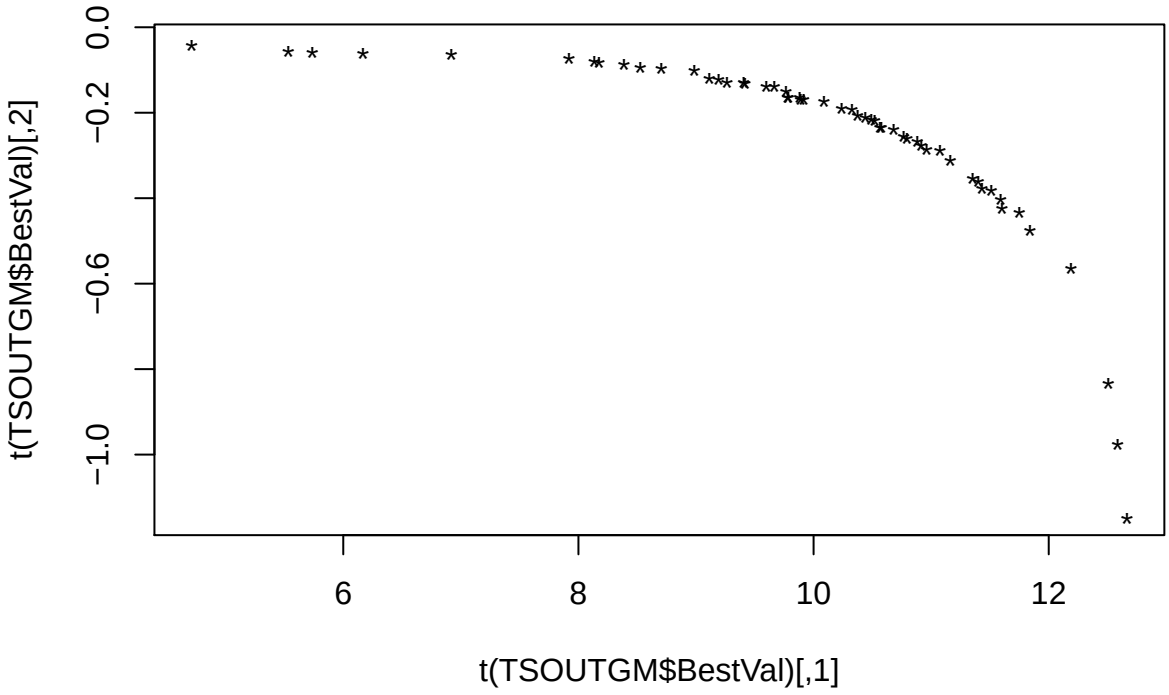
```
#

#### Two objectives, minimize inbreeding maximize gain
```

```
###selecting 30 mates,  setsizes=30
###Note Stat=BV_INB, nStat = 2,

BV_INB<-function(soln){
  DFMatessoln<-DFMates[soln,]
  P1<-factor(DFMatessoln[,1], levels=rownames(M1))
  P2<-factor(DFMatessoln[,2], levels=rownames(M2))
  P1<-model.matrix(~P1-1)
  P2<-model.matrix(~P2-1)
  P<-cbind(P1/2,P2/2)
  PKP<-mean(tcrossprod(P%*%K,P)) #inbreeding
  BVmatessoln<-mean(.5*(M1[DFMatessoln[,1],]%*%markereffects+
                          M2[DFMatessoln[,2],]%*%markereffects))# gain
  out<-c(BVmatessoln,-as.numeric(PKP))
  names(out)<-c("Gain","-Inb")
  return(out)
}



tsControl=TrainSelControl()
tsControl$npop=200
tsControl$niterations=30

TSOUTGM<-TrainSel(Candidates = list(1:nrow(DFMates)),
                  setsizes=10,
                  settypes = c("UOMS"),
                  Stat=BV_INB,
                  nStat = 2,
                  control=tsControl)
```

2 objectives, minimize inbreeding maximize gain

```
## Maximum number of iterations reached.
```

```
plot(t(TSOUTGM$BestVal), pch="*")
```




```
##dimension of the problem * number of solutions
dim(TSOUTGM$BestSol_int)
```

```
## [1] 10 54
```

```
########solution 1
head(DFMates[TSOUTGM$BestSol_int[,1],])
```

```
##                  Var1                    Var2
## 167       HBF0303-155 PUDMERICKA_CERVENOCLASA
## 167.1     HBF0303-155 PUDMERICKA_CERVENOCLASA
## 167.2     HBF0303-155 PUDMERICKA_CERVENOCLASA
## 282   BON_FERMIER_SVP                    1548
## 282.1 BON_FERMIER_SVP                    1548
## 305           PI94489                    1548
```

```
TSOUTGM$BestVal[,1]
```

```
## [1] 10.5769689 -0.2313157
```

```
########solution 2
head(DFMates[TSOUTGM$BestSol_int[,2],])
```

```
##                  Var1                    Var2
## 167       HBF0303-155 PUDMERICKA_CERVENOCLASA
## 167.1     HBF0303-155 PUDMERICKA_CERVENOCLASA
## 167.2     HBF0303-155 PUDMERICKA_CERVENOCLASA
## 282   BON_FERMIER_SVP                    1548
## 282.1 BON_FERMIER_SVP                    1548
```

```
## 897         SEG95-997                     I/23
```

```
TSOUTGM$BestVal[,2] ##, etc,...
```

```
## [1]  9.1139261 -0.1153726
```

```
##############################
 BV_INB_VAR<-function(soln){
   DFMatessoln<-DFMates[soln,]
   P1<-factor(DFMatessoln[,1], levels=rownames(M1))
   P2<-factor(DFMatessoln[,2], levels=rownames(M2))
   P1<-model.matrix(~P1-1)
   P2<-model.matrix(~P2-1)
   P<-cbind(P1/2,P2/2)
   varI<-mean(sapply(1:nrow(DFMatessoln), function(i){
     p1<-DFMatessoln[i,1]
     p2<-DFMatessoln[i,2]
     m1<-M1[p1,]
     m2<-M2[p2,]
     calculatecrossvalueM1(round(m1),round(m2),markereffects)
     #calculatecrossvalueM1(round(m1),round(m2),markereffects, markermap)

   }))

  PKP<-mean(tcrossprod(P%*%K,P)) #inbreeding
  BVmatessoln<-mean(.5*(M1[DFMatessoln[,1],]%*%markereffects+
                          M2[DFMatessoln[,2],]%*%markereffects))# gain


   out<-c(BVmatessoln, varI,-as.numeric(PKP))
   names(out)<-c("Gain", "Var","-Inb")
   return(out)

 }
```

```
####Three objectives, minimize inbreeding maximize gain and cross-variance
###selecting 10 mates, setsizes=10
###Note Stat=BV_INB_VAR, nStat = 3!

tsControl=TrainSelControl()
tsControl$npop=1000
tsControl$niterations=100

 TSOUT<-TrainSel(Candidates = list(1:nrow(DFMates)),
                 setsizes=100,
                 settypes = c("UOMS"),
                 Stat=BV_INB_VAR,
                 nStat = 3,
                 control = tsControl)
```

3 objectives, minimize inbreeding maximize gain and crossvariance

```
## Maximum number of iterations reached.
```

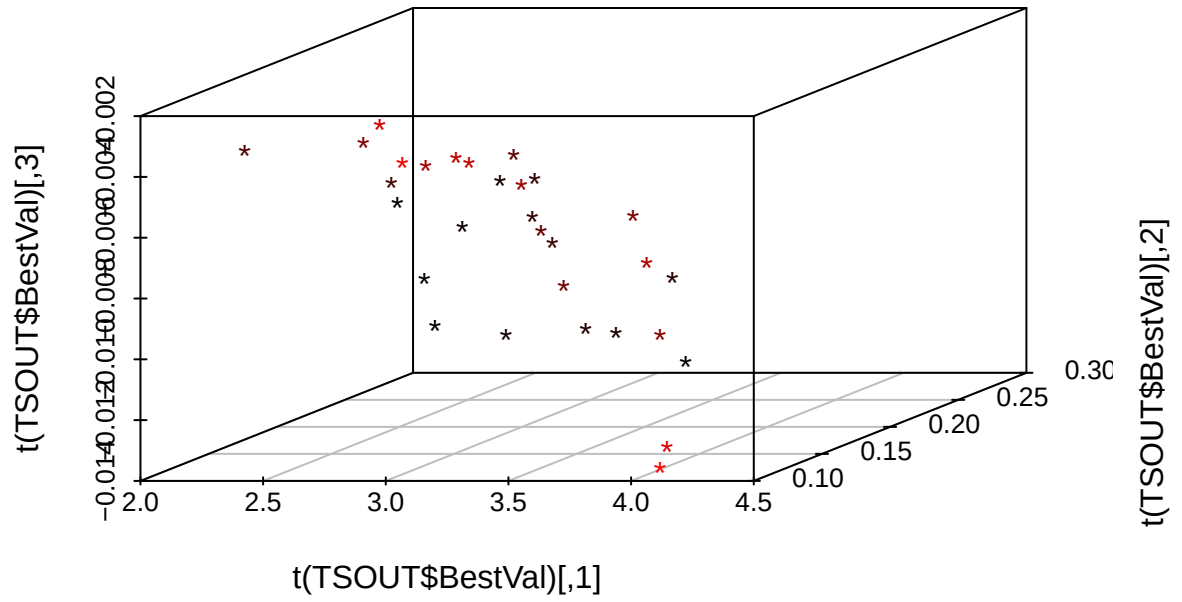
```
TSOUT$BestVal[,1:10] # values for first 10 results
```

```
##             [,1]         [,2]         [,3]         [,4]         [,5]
## [1,]  3.73439237  2.812470503  3.165532061  3.876932021  2.668326795
## [2,]  0.14925451  0.129350072  0.131257898  0.143362601  0.143381020
## [3,] -0.00608269 -0.002736918 -0.004019788 -0.009900633 -0.003562774
##              [,6]         [,7]         [,8]         [,9]        [,10]
## [1,]  3.336550135  3.428843052  2.515205884  2.624485742  3.316007563
## [2,]  0.139020796  0.153326820  0.215462971  0.171655460  0.157061522
## [3,] -0.004865906 -0.008467062 -0.009342331 -0.005397127 -0.006727785
```

```
ncol(TSOUT$BestVal) # Number of solutions found
```

```
## [1] 30
```

```
library(scatterplot3d)
sd3<-scatterplot3d(t(TSOUT$BestVal), pch="*",highlight.3d=TRUE)
```




```
########solution 1
head(DFMates[TSOUT$BestSol_int[,1],])
```

```
##           Var1                Var2
## 15  IWA8609184              SABINA
## 30      216/71              SABINA
## 32    N02Y5187              SABINA
## 69  IWA8609376      BON_FERMIER_SVP
## 93  IWA8602382 NEAR_ISOGENIC_(PM3B)
## 150     216/71       HAR_KOVSKAJA63
```

```
 TSOUT$BestVal[,1]
```

```
## [1]  3.73439237  0.14925451 -0.00608269
```

```
# ########solution 2
 head(DFMates[TSOUT$BestSol_int[,2],])
```

```
##               Var1                    Var2
## 1           SABINA                  SABINA
## 44  HAR_KOVSKAJA63         BON_FERMIER_SVP
## 50              MV4         BON_FERMIER_SVP
## 66            V/2-A         BON_FERMIER_SVP
## 147     IWA8611218          HAR_KOVSKAJA63
## 161         SABINA PUDMERICKA_CERVENOCLASA
```

```
 TSOUT$BestVal[,2]# ###,etc,...
```

```
## [1]  2.812470503  0.129350072 -0.002736918
```