Genomic Selection

Santos Barrera

4/15/2020

#1. Version control <https://github.com/Sybalemus1/Agro932Hw1.git>

#2. Place the following system of equations in matrix form and solve it using R. 5 X1 + 6 X2 = 3 3 X1 - 4 X2 = -6 Ax=y Ix=A(−𝟏)y x=A(−𝟏)y

#key matrix algebra c(): if scalar values are given as arguements, they are consider into a numeric vector matrix(): forms a matrix from the vector(s) of scalar calue given as arguments solve(X): returns the inverse of X A %\*% B: returns the product of matrices A and B t(X): retuns the transpose of X diag(X): when X is matrix, diag (X) returns a vector of the diagonal elements of X #Generate Matrix A and vector y

A <- matrix(c(5,6, 3,-4), byrow=T, ncol=2)  
A

## [,1] [,2]  
## [1,] 5 6  
## [2,] 3 -4

y <- matrix(c(3,-6), byrow=T)  
y

## [,1]  
## [1,] 3  
## [2,] -6

#Solution

x=solve (A) %\*% y  
x

## [,1]  
## [1,] -0.6315789  
## [2,] 1.0263158

A%\*%x

## [,1]  
## [1,] 3  
## [2,] -6

##3 GBLUP and RR-BLUP # Loblolly pine data

There are some accessions containing no phenotype. We need to remove these accessions first.

#geno\_file <- "https://jyanglab.com/img/data/Snp\_Data.csv"  
pheno <- read.csv("data/DATA\_rootnum\_age10\_rootnum.csv", header=TRUE, stringsAsFactors = FALSE)  
str(pheno)  
head(pheno)  
hist(pheno$Derregressed\_BV)  
geno <- read.csv("data/Snp\_Data.csv", header=TRUE, stringsAsFactors = FALSE)  
dim(geno)

### Remove missing phenotypes

sum(is.na(pheno$Derregressed\_BV))  
na.index <- which(is.na(pheno$Derregressed\_BV))  
head(na.index)  
# length(na.index)  
pheno <- pheno[-na.index, ]  
# Keep genotypes for these remaining lines  
geno <- geno[geno$Genotype %in% pheno$Genotype, ]  
# phenotypes   
y <- pheno$Derregressed\_BV  
y <- matrix(y, ncol=1)  
# markers   
geno <- geno[,-1] # 861 x 4853  
geno[geno == -9] <- NA

##SNP quality control In the geno matrix, row indicates individual, column indicates SNPs.

#Missingness and Minor Allele Frequency (MAF)

# missing rate  
missing <- apply(geno, 2, function(x){sum(is.na(x))/length(x)})  
hist(missing, breaks=100, col="blue", xlab="SNP Missing rate")  
# minor allele frequency  
maf <- apply(geno, 2, function(x){  
 frq <- mean(x, na.rm=TRUE)/2 # 1 allele  
 return(ifelse(frq > 0.5, 1-frq, frq))  
})  
hist(maf, breaks=100, col="blue", xlab="Minor Allele Freq")

Removing SNPs with high missing rate (missingness > 0.2) and low MAF (MAF < 0.05)

idx1 <- which(missing > 0.2) #159  
#length(idx1)  
idx2 <- which(maf < 0.05) #1642  
#length(idx2)  
idx <- unique(c(idx1, idx2)) #1784  
#length(idx)  
geno2 <- geno[, -idx]  
dim(geno2)

Missing marker imputation Using the mean value to infer the possible missed marker value

Z <- matrix(0, ncol=ncol(geno2), nrow=nrow(geno2))  
for (j in 1:ncol(geno2)){  
 #cat("j = ", j, '\n')  
 Z[,j] <- ifelse(is.na(geno2[,j]), mean(geno2[,j], na.rm=TRUE), geno2[,j])  
}  
sum(is.na(Z))

##Genomic Relationship SNP Matrix standardization

Zs <- scale(Z, center = TRUE, scale = TRUE)  
# dimensions   
n <- nrow(Zs) #923  
m <- ncol(Zs) #3069

Calcualte genomic relationship

Compute the second genomic relationship matrix of VanRaden (2008) using the entire markers. Then add a very small positive constant (e.g., 0.001) to the diagonal elements so that G matrix is invertible.

# Given matrices x and y as arguments, return a matrix cross-product. This is formally equivalent to (but usually slightly faster than) the call t(x) %\*% y (crossprod) or x %\*% t(y) (tcrossprod).  
G <- tcrossprod(Zs) / ncol(Zs)  
# G <- Zs %\*% t(Zs) /ncol(Zs)  
G <- G + diag(n)\*0.001

##Solve MME for GBLUP y=1μ+Zu+e

lambda <- 4.087 # fit$Ve / fit$Vm (lambda value obtained from rrBLUP Package)  
Ginv <- solve(G)  
ones <- matrix(1, ncol=1, nrow=n)  
Z <- diag(n)  
# Given matrices x and y as arguments, return a matrix cross-product. This is formally equivalent to (but usually slightly faster than) the call t(x) %\*% y (crossprod) or x %\*% t(y) (tcrossprod).  
LHS1 <- cbind(crossprod(ones), crossprod(ones, Z))   
LHS2 <- cbind(crossprod(Z, ones), crossprod(Z) + Ginv\*lambda)  
LHS <- rbind(LHS1, LHS2)  
RHS <- rbind( crossprod(ones, y), crossprod(Z,y) )  
sol <- solve(LHS, RHS)  
head(sol)  
tail(sol)

#Fit RR-BLUP by using the mixed.solve function in the

library(rrBLUP)  
fit <- mixed.solve(y = y, K=G)  
# marker additive genetic variance  
fit$Vu  
# residual variance  
fit$Ve  
# intercept   
fit$beta  
# marker additive genetic effects  
head(fit$u)  
tail(fit$u)  
# genomic h2  
fit$Vu / (fit$Vu + fit$Ve)  
# ratio of variance components   
fit$Ve / fit$Vu  
  
# plot(x=sol2[-1], y=fit2$u)

##RR-BLUP Effect of the genomewide markers y=1b+Zm+e Manual calculation

lambda <- 4326.212 # fit$Ve / fit$Vu (lambda value obtained from rrBLUP Package)  
ones <- matrix(1, ncol=1, nrow=n)  
I <- diag(m)  
LHS1 <- cbind(crossprod(ones), crossprod(ones, Zs))   
LHS2 <- cbind(crossprod(Zs, ones), crossprod(Zs) + I\*lambda)  
LHS <- rbind(LHS1, LHS2)  
RHS <- rbind( crossprod(ones, y), crossprod(Zs,y) )  
sol2 <- solve(LHS, RHS)  
head(sol2)  
tail(sol2)  
eff <- sol2[-1]  
head(eff)  
plot(1:length(eff), eff, pch=16)

![A screenshot of a cell phone

Description automatically generated]()

#RR-Blup Using rrBLUP package

library(rrBLUP)  
fit2 <- mixed.solve(y = y, Z=Zs)  
# marker additive genetic variance  
fit2$Vu  
# residual variance  
fit2$Ve  
# intercept   
fit2$beta  
# marker additive genetic effects  
head(fit2$u)  
tail(fit2$u)  
# ratio of variance components   
fit2$Ve / fit2$Vu  
plot(x=sol2[-1], y=fit2$u)

![A screenshot of a map

Description automatically generated]()

#K-fold validation GBLUP k=10 folds I don’t know how to divide the data in sets of 10 I did the same as you did in the class

n.trn <- 600  
n.tst <- 325  
y.trn <- y[1:n.trn]  
y.tst <- y[n.trn+1:n.tst]  
Zs.trn <- Zs[1:n.trn,]  
Zs.tst <- Zs[n.trn+1:n.tst,]  
  
Gtrn <- tcrossprod(Zs.trn) / ncol(Zs.trn)  
Gtrn <- Gtrn + diag(n.trn)\*0.001  
Gtst.trn <- tcrossprod(Zs.tst, Zs.trn) / ncol(Zs.tst)  
#Gtrn <- G[1:n.trn, 1:n.trn]  
#Gtst.trn <- G[n.trn+1:n.tst, 1:n.trn]  
  
lambda <- 1.348411 # fit$Ve / fit$Vu  
Ginv.trn <- solve(Gtrn)  
ones <- matrix(1, ncol=1, nrow=n.trn)  
Z <- diag(n.trn)  
LHS1 <- cbind(crossprod(ones), crossprod(ones, Z))   
LHS2 <- cbind(crossprod(Z, ones), crossprod(Z) + Ginv.trn\*lambda)  
LHS <- rbind(LHS1, LHS2)  
RHS <- rbind( crossprod(ones, y.trn), crossprod(Z,y.trn) )  
sol.trn <- solve(LHS, RHS)  
  
# prediction  
y.hat <- Gtst.trn %\*% Ginv.trn %\*% matrix(sol.trn[c(2:(n.trn+1))])  
GBLUP.trn <- cor(y.hat, y[(n.trn+1):n]) #correlation coeficient 0.305 = 30% can be pedicted   
# plot(y.hat, y[(n.trn+1):n])

#K-fold validation RR-BLUP k=10 folds I dont know how to divide the data in sets of 10 I did the same as you did in the class

Zs.trn <- Zs[1:n.trn, ]  
Zs.tst <- Zs[n.trn+1:n.tst, ]  
lambda <- 4326.212 # fit$Ve / fit$Vu  
ones <- matrix(1, ncol=1, nrow=n.trn)  
I <- diag(m)  
LHS1 <- cbind(crossprod(ones), crossprod(ones, Zs.trn))   
LHS2 <- cbind(crossprod(Zs.trn, ones), crossprod(Zs.trn) + I\*lambda)  
LHS <- rbind(LHS1, LHS2)  
RHS <- rbind( crossprod(ones, y.trn), crossprod(Zs.trn, y.trn) )  
sol.trn <- solve(LHS, RHS)  
  
# prediction  
y.hat2 <- Zs.tst %\*% matrix(sol.trn[-1])  
RRBLUP.trn <- cor(y.hat2, y[(n.trn+1):n])  
# plot(y.hat2, y[(n.trn+1):n])

### 4. Visualize the prediction accuracy results using the box plot or violin plot.

library(ggplot2)  
plot\_acc <- data.frame(accuracy= c(GBLUP.trn, RRBLUP.trn),method = c(rep("GBLUP", 10), rep("rrBLUP", 10)) )  
plot <- ggplot(plot\_acc, aes(x=method, y=accuracy, fill=method)) +   
 geom\_violin(trim=FALSE)+  
 labs(title="Prediction Accuracy by GBLUP and rrBLUP", x="", y = "Prediction Accuracy")+  
 geom\_boxplot(width=0.1, fill="white")+  
 scale\_fill\_brewer(palette="reds") +   
 theme\_classic()  
plot

![A close up of a logo

Description automatically generated]()