Genomic Selection

Jinliang Yang

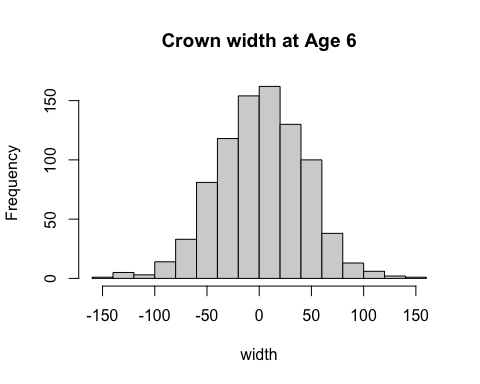
04-24-2025

## Path Normalization

# A real world example: Loblolly pine data

In this example, we will use the breeding values of crown width across the planting beds at age 6 (CWAC6).

# read phenotype and SNP files  
pheno\_file <- "https://jyanglab.com/img/data/DATA\_nassau\_age6\_CWAC.csv"  
geno\_file <- "https://jyanglab.com/img/data/Snp\_Data.csv"  
  
pheno <- read.csv(pheno\_file, header=TRUE, stringsAsFactors = FALSE)  
hist(pheno$Derregressed\_BV, main="Crown width at Age 6", xlab="width")



# geno[1:10, 1:10]

# Loblolly pine data

### Remove missing phenotypes

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

na.index <- which(is.na(pheno$Derregressed\_BV))  
# length(na.index)  
pheno <- pheno[-na.index, ]  
  
# phenotypes   
y <- pheno$Derregressed\_BV  
y <- matrix(y, ncol=1)

# Genotype data: SNP quality control

In the geno matrix, row indicates individual, column indicates SNPs.

### Missingness and MAF

geno <- read.csv(geno\_file, header=TRUE, stringsAsFactors = FALSE)  
dim(geno)

## [1] 926 4854

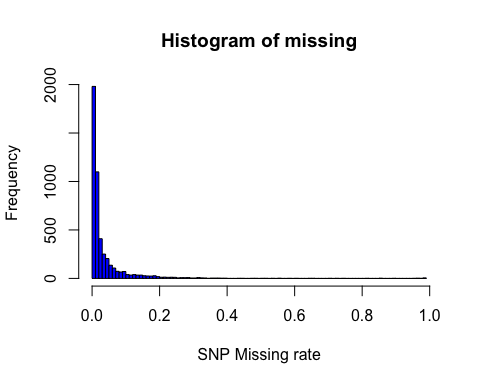
# Keep genotypes for these remaining lines  
geno <- geno[geno$Genotype %in% pheno$Genotype, ]  
# markers   
geno <- geno[,-1] # 861 x 4853  
geno[geno == -9] <- NA  
  
# missing rate  
missing <- apply(geno, 2, function(x){sum(is.na(x))/length(x)})  
# minor allele frequency  
maf <- apply(geno, 2, function(x){  
 frq <- mean(x, na.rm=TRUE)/2 # 1 allele  
 return(ifelse(frq > 0.5, 1-frq, frq))  
})

# Genotype data: SNP quality control

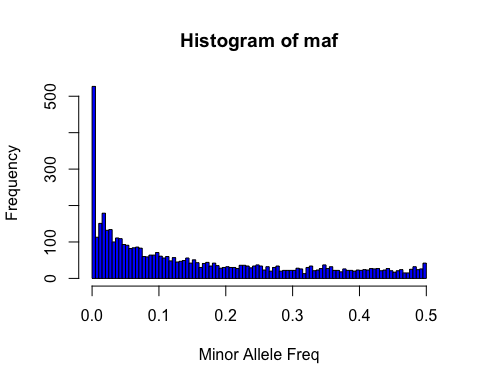
In the geno matrix, row indicates individual, column indicates SNPs.

#### Plot the results

hist(missing, breaks=100, col="blue", xlab="SNP Missing rate")



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)

* Question: How many markers are removed?

idx1 <- which(missing > 0.2) #154  
idx2 <- which(maf < 0.05) #1647  
idx <- unique(c(idx1, idx2)) #1784  
  
geno2 <- geno[, -idx]  
dim(geno2)

## [1] 861 3069

### Missing marker imputation

Replace missing marker genotypes with **mean values**. Then store the marker genotypes in a matrix object Z.

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))  
write.table(Z, "data/Z.txt", sep="\t", row.names = FALSE,   
 col.names=FALSE, quote=FALSE)

# Genomic relationship

### SNP Matrix standardization

Standardize the genotype matrix to have a mean of zero and variance of one. Save this matrix as Zs.

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

### Calcualte genomic relationship

* Compute the second genomic relationship matrix of [VanRaden (2008)](https://www.ncbi.nlm.nih.gov/pubmed/18946147) 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. A singular matrix cannot be inverted, which poses computational challenges.

# 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

* tcrossprod(Zs) computes the matrix product of Zs and its transpose, yielding an n x n matrix where n is the number of individuals.
* Dividing by the number of markers (ncol(Zs)) standardizes the matrix.

# Solve MME for GBLUP

Set up mixed model equations (MME) by fitting the model:

* where is the intercept,
* is the incident matrix of individuals,
* is the breeding value of the individuals,
* and is the residual.

Directly take the inverse of LHS to obtain the solutions for GBLUP. Report the estimates of intercept and additive genetic values. Use .

lambda <- 1.35 # fit$Ve / fit$Vg  
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)

## [,1]  
## [1,] 2.275528  
## [2,] 12.915583  
## [3,] -15.949010  
## [4,] 18.411816  
## [5,] 4.649033  
## [6,] -23.828528

tail(sol)

## [,1]  
## [857,] -3.877303  
## [858,] 5.900186  
## [859,] 7.631312  
## [860,] -49.125424  
## [861,] -8.490103  
## [862,] -37.223103

# R package: rrBLUP

Fit GBLUP by using the mixed.solve function in the [rrBLUP](https://cran.r-project.org/web/packages/rrBLUP/index.html) R package.

* Report the estimates of intercept and additive genetic values.
* Do they agree with previous estimates?
* Also, report the estimated genomic heritability and the ratio of variance components .

#install.packages("rrBLUP")  
library(rrBLUP)  
fit <- mixed.solve(y = y, K=G)  
# additive genetic variance  
fit$Vu

## [1] 721.3393

# residual variance  
fit$Ve

## [1] 997.0729

# intercept   
fit$beta

## [1] 2.275528

# additive genetic values  
head(fit$u)

## [1] 12.872001 -16.009856 18.153310 4.307152 -23.873051 16.372003

tail(fit$u)

## [1] -3.964626 6.047412 7.634191 -48.812717 -8.437586 -36.961484

# genomic h2  
fit$Vu / (fit$Vu + fit$Ve)

## [1] 0.4197708

# ratio of variance components   
fit$Ve / fit$Vu

## [1] 1.382252

# plot(x=sol[-1], y=fit$u)

# RR-BLUP

Set up mixed model equations (MME) by fitting the model , where is the intercept, is the standardized marker genotypes (Zs), is the additive marker genetic effects, and is the residual.

Directly take the inverse of LHS to obtain the solutions for marker-based GBLUP (RR-BLUP). Report the estimates of intercept and marker additive genetic effects. Use .

–

lambda <- 4326.212 # fit$Ve / fit$Vu  
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)

## [,1]  
## [1,] 2.27552828  
## [2,] 0.25984118  
## [3,] -0.03032116  
## [4,] -0.12452689  
## [5,] 0.15758584  
## [6,] -0.13104812

tail(sol2)

## [,1]  
## [3065,] 0.009042828  
## [3066,] 0.065547947  
## [3067,] -0.235825005  
## [3068,] 0.042984822  
## [3069,] 0.102930180  
## [3070,] 0.262549987

# Use rrBLUP package

Fit RR-BLUP by using the mixed.solve function in the [rrBLUP](https://cran.r-project.org/web/packages/rrBLUP/index.html) R package.

* Report the estimates of intercept and marker additive genetic effects.
* o they agree with the estimates with the manual calculation?
* Also, report the ratio of variance components .

library(rrBLUP)  
fit2 <- mixed.solve(y = y, Z=Zs)  
# marker additive genetic variance  
fit2$Vu

## [1] 0.2350402

# residual variance  
fit2$Ve

## [1] 997.7947

# intercept   
fit2$beta

## [1] 2.275528

# marker additive genetic effects  
head(fit2$u)

## [1] 0.26285584 -0.03075328 -0.12570117 0.16019719 -0.13267752 -0.06454280

tail(fit2$u)

## [1] 0.008232377 0.066259519 -0.237935580 0.042789624 0.103950959  
## [6] 0.264838013

# ratio of variance components   
fit2$Ve / fit2$Vu

## [1] 4245.208

# plot(x=sol2[-1], y=fit2$u)

# K-fold validation

Repeat GBLUP but treat the first 600 individuals as a training set and predict the additive genetic values of the remaining individuals in the testing set. - What is the predictive correlation in the testing set? Use .

n.trn <- 600  
n.tst <- 261  
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))])  
cor(y.hat, y[(n.trn+1):n])

## [,1]  
## [1,] 0.4635443

# plot(y.hat, y[(n.trn+1):n])

Repeat RR-BLUP but treat the first 600 individuals as a training set and predict the additive genetic values of the remaining individuals in the testing set. - What is the predictive correlation in the testing set? Use . - Also, compare this predictive correlation to the one from GBLUP.

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])  
cor(y.hat2, y[(n.trn+1):n])

## [,1]  
## [1,] 0.4635768

plot(y.hat2, y[(n.trn+1):n])

