Statistical genetics exercises

Vincent Garin

2024-02-22

### Elements of probability

#### Ex 1

The probability of a recombination between two region of the genome and separated by 1 centiMorgan can be modeled by a Poisson distribution. Let us assume that after analyzing all the data available on in your laboratory on your favourite species, you could observe on average 0.7 recombination over such a segment. In your next analysis you discover that on several point of your genome you have 2 recombination per centiMorgan. What is the probability of such an observation.

#### Answer

dpois(x = 2, lambda = 0.7)

## [1] 0.1216634

((0.7^2) / 2) \* exp(-0.7)

## [1] 0.1216634

#### Ex 2

Using the cumulative distribution function, calculate the probability that a random variable distributed with a distribution fall between -1.96 and 1.96.

pnorm(q = 1.96, mean = 0, sd = 1) - pnorm(q = -1.96, mean = 0, sd = 1)

## [1] 0.9500042

### Matrix

#### Ex 1

Calculate the following matrix with

#### Answer

library(matrixcalc)  
A <- matrix(c(1, 2, 0, 3), 2)  
t(A)

## [,1] [,2]  
## [1,] 1 2  
## [2,] 0 3

solve(t(A))

## [,1] [,2]  
## [1,] 1 -0.6666667  
## [2,] 0 0.3333333

solve(t(A)) %\*% solve(t(A))

## [,1] [,2]  
## [1,] 1 -0.8888889  
## [2,] 0 0.1111111

matrix.power(solve(t(A)), 2)

## [,1] [,2]  
## [1,] 1 -0.8888889  
## [2,] 0 0.1111111

#### Ex 2

1. Using the provided marker matrix , calculate a kinship matrix using the van Raden method , where is the centered marker matrix with and frequency of the marker.
2. Using the microbenchmark package compare the speed of inversion of the kinship matrix using standard function slove() and cholesky decomposition cholinv2(). Sometimes, the inversion of a kinship matrix is problematic because it is singular. It is possible to solve the problem by adding a small digit (e.g. 10E-4) to the diagonal.

#### Answer

# Kinship matrix construction  
library(AGHmatrix)  
load("./data/geno.RData")  
X <- geno  
n <- nrow(X)  
p <- apply(X, 2, FUN = function(x) sum(x)/(2\*length(x)))  
P <- rep(1, nrow(X)) %\*% (2\*t(p))   
Z <- X - P  
m <- ncol(X)  
D <- diag(1/(m \* 2 \* p \* (1 - p)))  
ZZt <- tcrossprod(Z)  
den <- 2 \* sum(p \* (1 -p))  
K <- ZZt/den  
K2 <- Gmatrix(SNPmatrix = X, method = "VanRaden")

## Initial data:   
## Number of Individuals: 355   
## Number of Markers: 5000   
##   
## Missing data check:   
## Total SNPs: 5000   
## 0 SNPs dropped due to missing data threshold of 0.5   
## Total of: 5000 SNPs   
##   
## MAF check:   
## No SNPs with MAF below 0   
##   
## Heterozigosity data check:   
## No SNPs with heterozygosity, missing threshold of = 0   
##   
## Summary check:   
## Initial: 5000 SNPs   
## Final: 5000 SNPs ( 0 SNPs removed)   
##   
## Completed! Time = 1.04 seconds

# Matrix inversion  
library(microbenchmark)  
  
# K <- solve(K)  
  
lda <- diag(rep(0.00001, n))  
  
K <- K + lda  
  
microbenchmark(K\_i <- solve(K), K\_i <- chol2inv(K), times = 20)

## Unit: milliseconds  
## expr min lq mean median uq max neval  
## K\_i <- solve(K) 21.7088 22.8956 24.428505 23.84955 24.78195 32.4593 20  
## K\_i <- chol2inv(K) 5.0649 5.3256 6.050285 5.51675 6.12535 13.6047 20  
## cld  
## a   
## b

#### Ex 3

1. It is possible to express a kinship symmetric matrix using the eigen decomposition . Using the fact that is an orthogonal matrix with the following property , show that .

Calculate in R the inverse of the kinship using

1. To speed-up the computation, it is convenient to only select the ‘top’ principal component that correspond to a certain value (e.g. ). Perform the eigen decomposition on the K matrix and select only the top PC that correspond to 0.995. What is the rank reduction?
2. Calculate a simple G-BLUP model using the rrBLUP package and the first phenotype. to compare the estimation of the genetic variance. You can use the following code

# m <- kin.blup(data = d, geno = "GID", pheno = "y", K = K\_red)  
# m$Vg

#### Answer

#### A.

E <- eigen(K)  
U <- E$vectors  
U\_i <- solve(U)  
Ut <- t(U)  
D <- diag(E$values)  
  
K\_i <- U %\*% solve(D) %\*% Ut  
K\_i2 <- solve(K)

#### Eigen value decomposition (PCA) Intermezzo

The eigenvalue decomposition allows to synthesis the information contained in a (squared) matrix by forming linear combination of the variable (Principal components).

load("./data/data\_UScrime.RData")  
data <- UScrime  
data <- cor(data)  
  
E <- eigen(x = data)

The eigenvectors are the loadings . The matrix multiplied by the loadings give the PC. The PC are formed as such that they are orthogonal. The eigen vectors are also orthogonal

U <- E$vectors  
PC <- t(as.matrix(data)) %\*% U  
colnames(PC) <- paste0('PC', 1:ncol(data))  
  
crossprod(PC[, 1], PC[, 2])

## [,1]  
## [1,] -1.332268e-15

t(PC[, 1]) %\*% PC[, 3]

## [,1]  
## [1,] -7.605028e-15

t(PC[, 2]) %\*% PC[, 3]

## [,1]  
## [1,] -3.413936e-15

crossprod(U[, 1], U[, 2])

## [,1]  
## [1,] -4.857226e-17

t(U[, 1]) %\*% U[, 3]

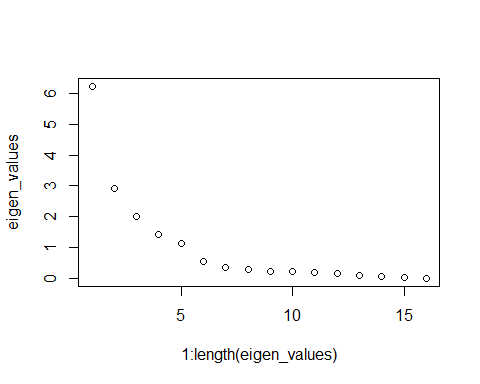
## [,1]  
## [1,] 1.94289e-16

t(U[, 2]) %\*% U[, 3]

## [,1]  
## [1,] 8.673617e-17

The eigenvalues are proportional to the amount of data transformation/elogation due to the eigen vector.

eigen\_values <- E$values  
plot(y = eigen\_values, x = 1:length(eigen\_values))



#### B.

E <- eigen(K)  
PC <- K %\*% E$vectors  
V\_PC <- apply(PC, 2, var)  
V\_prop <- V\_PC/(sum(V\_PC))  
sum(V\_prop)

## [1] 1

rk <- sort(which(cumsum(V\_prop) > 0.995))[1]  
K\_red <- E$vectors[, 1:rk] %\*% diag(E$values[1:rk]) %\*% t(E$vectors[, 1:rk])

#### C.

load(file = "./data/pheno.RData")  
y <- pheno$SB1\_PH  
GID <- rownames(pheno)  
d <- data.frame(y = y,GID)  
d$GID <- factor(d$GID)  
rownames(K\_red) <- colnames(K\_red) <- GID  
  
library(rrBLUP)  
m <- kin.blup(data = d, geno = "GID", pheno = "y", K = K\_red)  
m$Vg

## [1] 751.5881

m2 <- kin.blup(data = d, geno = "GID", pheno = "y", K = K)  
m2$Vg

## [1] 790.587

### Linear model

#### Ex1

Given the expression of the predicted values as and the expression of , show that

#### Answer

#### Ex2: Function for OLS estimation

Create your own OLS estimate function. Test it using the iris data to calculate the following model . The model contain no general intercept but one intercept per species. compare with the lm() function.

#### Answer

data("iris")  
  
y <- iris$Sepal.Length  
X <- model.matrix(~ -1 + iris$Species)  
X <- cbind(X, iris$Petal.Length)  
  
colnames(X) <- c("setosa", "versicolor", "virginica", "Petal.Length")  
  
OLS <- function(y, X){  
 Beta <- solve(crossprod(X)) %\*% crossprod(X, y)  
 Beta  
}  
  
B\_OLS <- OLS(y, X)  
  
m <- lm(Sepal.Length ~ -1 + Species + Petal.Length, data = iris)  
  
B\_OLS

## [,1]  
## setosa 3.6835266  
## versicolor 2.0825548  
## virginica 1.5658574  
## Petal.Length 0.9045646

m$coefficients

## Speciessetosa Speciesversicolor Speciesvirginica Petal.Length   
## 3.6835266 2.0825548 1.5658574 0.9045646

#### Ex3: Regression on parents

Different methods exist to calculate the heritability. A famous method is the parent offspring regression.

In the abscence of environmental effect, it can be shown that

(for single parent regression)

(for average or mid-parent regression)

Using the Galton data that can be found there <https://ytliu0.github.io/Stat390EF-R-Independent-Study-archive/RMarkdownExercises/Galton.txt>

1. Calculate the heritability using single parent regression
2. Calculate the heritability using mid parent regression

#### Answer

data <- read.table(file = "https://ytliu0.github.io/Stat390EF-R-Independent-Study-archive/RMarkdownExercises/Galton.txt", h = TRUE)  
  
# regression single parent  
m1 <- lm(Height ~ Mother, data = data)  
  
h2\_s <- 2\*m1$coefficients[2]  
  
mid\_parent <- 0.5\*(data$Father + data$Mother)  
  
m2 <- lm(data$Height ~ mid\_parent)  
  
h2\_m <- m2$coefficients[2]

### Generalized least square

#### Ex 1: Fast GWAS with approximate GLS estimate

Let us remember that . A popular method to increase the speed of GWAS computation is to obtain an estimation of variance copmpnents ( and ) using the following model

where,

and is a kinship matrix

The estimation of and allow to reconstruct (The VCOV of the model) without QTL term, and then to insert in the formula to estimate the value of the QTL at each position, which will give us an approximation of the following model

where is the QTL term and is the QTL effect. The estimation of the significance of allows to build a Manhattan plot.

1. Derive the variance expression of
2. using the example geno and pheno data estimate the variance components and using the following code from the **regress** package

library(regress)  
library(rrBLUP)  
  
load("./data/geno\_GWAS.RData")  
load("./data/pheno\_GWAS.RData")  
load("./data/map\_GWAS.RData")  
  
y <- pheno$res  
# for simplicity center the phenotype data  
y <- y - mean(y)  
X <- geno  
  
# calculate the Kinship matrix  
K <- A.mat(X = geno)  
  
m <- regress(formula = y ~ 1, Vformula = ~ K)  
summary(m)  
  
# Extract the V estimates  
# Sg <- c()  
# Se <- c()

1. Using the and estimates, reconstruct the VCOV of the model derive in 1. Chekc that the inverse of this matrix is the same as the one returned by regress()

V\_i\_reg <- m$W  
  
# this expression should only give 0  
V\_i\_rec - V\_i\_reg

1. Using a single marker position (406) calculate the Beta GLS using the given formula and the estimated , as well as the significance of this value after calculating the corresponding F-statistic

According to the literature, a general statistical F-test for take the form

In our case, since we centered the data (), we only test 1 parameter (, QTL term), so and the F-statistic expression reduce to

The F statistics is distributed with and . You can obtain the p-value using

pf(q = F\_stat, df1 = 1, df2 = n - 1, lower.tail = FALSE)

1. Repeat the previous operation for all the genetic markers using a loop. (advice: use the function tryCatch() to prevent the error in both and calculation).

Plot the results using the following code

d <- map[, c(2, 4)]  
d$log10pval <- -log10(F\_pval)  
  
p <- ggplot(data = d, aes(x = pos.cM, y = log10pval)) + geom\_point() +  
 facet\_wrap(~chr, nrow = 1)

1. Compare the results you obtain with the standard GWAS functio from **rrBLUP**

# comparison with GWAS function from rrBLUP package  
g\_data <- data.frame(map[, -3], t(geno), check.names = FALSE)  
  
# Classical GWAS with K (kinship correction)  
Q\_GWAS <- GWAS(pheno = pheno[, -3], geno = g\_data, K = K)

##### Answer

1. Calculation of the variance estimate

library(regress)  
library(rrBLUP)  
  
load("./data/geno\_GWAS.RData")  
load("./data/pheno\_GWAS.RData")  
load("./data/map\_GWAS.RData")  
  
y <- pheno$res  
# for simplicity center the phenotype data  
y <- y - mean(y)  
X <- geno  
  
# calculate the Kinship matrix  
K <- A.mat(X = geno)  
  
m <- regress(formula = y ~ 1, Vformula = ~ K)  
summary(m)

## Likelihood kernel: K = (Intercept)  
##   
## Maximized log likelihood with kernel K is -107.965   
##   
## Linear Coefficients:  
## Estimate Std. Error  
## (Intercept) 0 0.16  
##   
## Variance Coefficients:  
## Estimate Std. Error  
## K 0.604 0.362  
## In 2.532 0.526

# Extract the V estimates  
Sg <- m$sigma[1]  
Se <- m$sigma[2]

1. Reconstruction of the VCOV

GID <- as.factor(rownames(K))  
Z <- model.matrix(~ -1 + GID)  
  
# build the VCOV matrix and its inverse  
V <- Z %\*% K %\*% t(Z) \* Sg + (diag(nrow(K))\*Se)  
V\_i <- solve(V)  
  
# check that regress give the same  
V\_i\_reg <- m$W  
# V\_i - V\_i\_reg

1. Calculate the Beta GLS and F statistic at a single position

X\_i <- X[, 406, drop = FALSE]  
  
B\_i <- solve(t(X\_i) %\*% V\_i %\*% X\_i) %\*% (t(X\_i) %\*% V\_i %\*% y)  
  
F\_i <- t(B\_i) %\*% solve(solve(t(X\_i) %\*% V\_i %\*% X\_i)) %\*% (B\_i)  
  
F\_pval <- pf(q = F\_i, df1 = 1, df2 = nrow(X\_i)-1, lower.tail = FALSE)

1. Calculate the B\_GLS and significance for all positions

library(ggplot2)  
# space to store the values  
F\_pval <- rep(NA, ncol(X))  
  
for(i in 1:ncol(X)){  
   
 X\_i <- X[, i, drop = FALSE]  
   
 B\_i <- tryCatch((solve(t(X\_i) %\*% V\_i %\*% X\_i)) %\*% (t(X\_i) %\*% V\_i %\*% y),  
 error = function(x) NA)  
   
 F\_i <- tryCatch(t(B\_i) %\*% solve(solve(t(X\_i) %\*% V\_i %\*% X\_i)) %\*% (B\_i)/1,  
 error = function(x) NA)  
   
 F\_pval[i] <- pf(q = F\_i, df1 = 1, df2 = nrow(X\_i), lower.tail = FALSE)  
   
}  
  
d <- map[, c(2, 4)]  
d$log10pval <- -log10(F\_pval)  
  
p <- ggplot(data = d, aes(x = pos.cM, y = log10pval)) + geom\_point() +  
 facet\_wrap(~chr, nrow = 1)

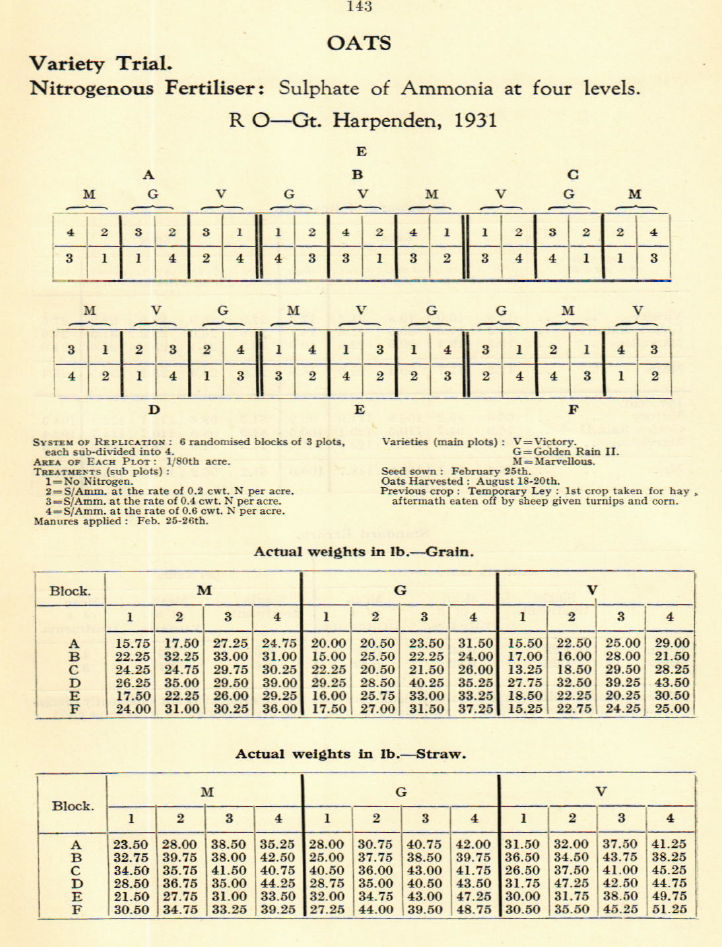
1. Comparison with GWAS from **rrBLUP**

# comparison with GWAS function from rrBLUP package  
g\_data <- data.frame(map[, -3], t(geno), check.names = FALSE)  
  
# Classical GWAS with K (kinship correction)  
Q\_GWAS <- GWAS(pheno = pheno[, -3], geno = g\_data, K = K)

### Mixed model

#### Ex1 Phenotype analysis Oats experiment

1. Get the Oats experiment data using the R code below. It is a split-plot design with varieties considered as the main plot and fertilization (nitrogen) considered as the sub-plot.



Original report Oats experiment (1935)

library(regress)  
library(nlme)  
library(agridat)  
library(dplyr)

##   
## Attachement du package : 'dplyr'

## L'objet suivant est masqué depuis 'package:nlme':  
##   
## collapse

## Les objets suivants sont masqués depuis 'package:stats':  
##   
## filter, lag

## Les objets suivants sont masqués depuis 'package:base':  
##   
## intersect, setdiff, setequal, union

library(desplot)

## Warning: le package 'desplot' a été compilé avec la version R 4.3.3

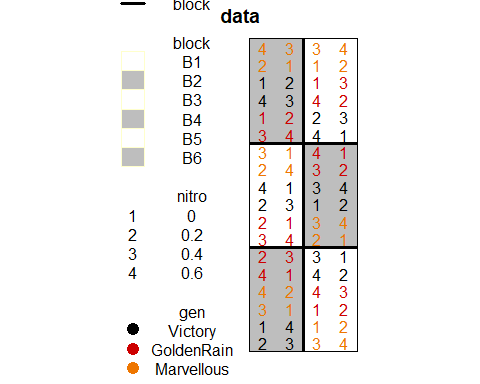
library(lme4)

## Le chargement a nécessité le package : Matrix

##   
## Attachement du package : 'lme4'

## L'objet suivant est masqué depuis 'package:nlme':  
##   
## lmList

# Oats data experimental design  
data(yates.oats)  
data <- yates.oats  
names(Oats) <- c("B","V","N","Y")  
  
# reorganise the data  
data$gen <- factor(as.character(data$gen),  
 levels = c("Victory", "GoldenRain", "Marvellous"))  
data <- data %>% arrange(block, gen, nitro)  
  
  
# reorganise column  
data <- data %>% select(row, col, block, gen, nitro, grain, straw, yield)  
data$yield2 <- Oats$Y  
  
# order the block and nitrogen (not essential)  
data$block <- ordered(data$block, levels = unique(data$block))  
data$nitro <- ordered(data$nitro, levels = unique(data$nitro))  
  
# Experiment design  
desplot(data, block ~ col\*row,  
 col.regions=c("white", "grey"),  
 out1=block, num=nitro, col=gen,  
 cex=1, aspect=511/176)



# change variable names  
names(data) <- c("R", "C","B","V","N","GR","ST", "YLD", "Y")

1. Propose a mixed model to analyse those data. Keep in mind that controlling the error at the different level of the split-plot design is a central component of the analysis of those designs.
2. Fit your model using one of the following package (regress, or nlme).
3. Get the BLUE (e.g of variety) and the BLUP (e.g. of the block) estimates. You can use the function BLUP() (regress) or ranef() (nlme).
4. “Manual” computation of the mixed model estimates using the Henderson equations. Look at the code below, try to identify the element of the MM equation. Compare the estimates with what you obtained before.

# manual computation of the BLUP using MM formulas  
y <- data$Y  
N <- nrow(data)  
I <- diag(N)  
X <- model.matrix(~ -1 + N + V, data = data)  
ZB <- model.matrix(~ -1 + B , data = data)  
ZBV <- model.matrix(~ -1 + V:B , data = data)  
  
# Possibility to get the incidences matrix from  
# regress output  
ZBV\_reg <- m\_reg$Z$`I(B:V)`  
  
sB <- m\_reg$sigma[1]  
sBV <- m\_reg$sigma[2]  
sE <- m\_reg$sigma[3]  
  
lbd\_B <- sE/sB  
lbd\_BV <- sE/sBV  
  
XX <- crossprod(X, X)  
XZB <- t(X) %\*% ZB  
ZBX <- t(ZB) %\*% X  
XZBV <- t(X) %\*% ZBV  
ZBVX <- t(ZBV) %\*% X  
  
ZBZB <- crossprod(ZB, ZB)  
ZBZBV <- crossprod(ZB, ZBV)  
  
ZBVZBV <- crossprod(ZBV, ZBV)  
ZBVZB <- crossprod(ZBV, ZB)  
  
ZZ\_B <- ZBZB + (lbd\_B \* diag(nrow(ZBZB)))  
ZZ\_BV <- ZBVZBV + (lbd\_BV \* diag(nrow(ZBVZBV)))  
  
# MM equations  
C <- rbind(cbind(XX, XZB, XZBV),  
 cbind(ZBX, ZZ\_B, ZBZBV),  
 cbind(ZBVX, ZBVZB, ZZ\_BV))  
rhs <- rbind(t(X) %\*% y, t(ZB) %\*% y, t(ZBV) %\*% y)  
BU <- solve(C) %\*% rhs  
  
# compare with the estimates returned by regress  
BU\_reg <- c(m\_reg$beta, BLUP\_reg\_vec)  
BU\_comp <- data.frame(MM\_est = BU, reg\_est = BU\_reg)

#### Answer

#### 2.

We could assume the following mixed model

We consider and as fixed because there are few levels that are clearly identifiable and we are interested ind knowing the estimates of those effects.

and are the blocking structure inside which randomisation was performed. Therefore, they are design parameter for which we want to control for the potential “nuisance” effect.

We also want to control for the fact that observation contained in the same randomisation unite are correlated.

#### 3.

# fit the model:  
# yield = nitrogen + variety + block(r) + block:variety(r)   
  
## Using regress  
m\_reg <- regress(Y~-1+N+V,~B+I(B:V),identity=TRUE,verbose=0,data=data)  
summary(m\_reg)

## Likelihood kernel: K = N0+N0.2+N0.4+N0.6+VGoldenRain+VMarvellous  
##   
## Maximized log likelihood with kernel K is -214.975   
##   
## Linear Coefficients:  
## Estimate Std. Error  
## N0 73.042 8.220  
## N0.2 92.542 8.220  
## N0.4 107.875 8.220  
## N0.6 117.042 8.220  
## VGoldenRain 6.875 7.079  
## VMarvellous 12.167 7.079  
##   
## Variance Coefficients:  
## Estimate Std. Error  
## B 214.468 168.794  
## I(B:V) 109.700 67.741  
## In 162.558 32.189

## Using lme (nlme)  
m\_lme <- lme(Y~-1+N+V,random=~1|B/V,data=data,method="REML")  
summary(m\_lme)

## Linear mixed-effects model fit by REML  
## Data: data   
## AIC BIC logLik  
## 586.0688 605.7756 -284.0344  
##   
## Random effects:  
## Formula: ~1 | B  
## (Intercept)  
## StdDev: 14.64488  
##   
## Formula: ~1 | V %in% B  
## (Intercept) Residual  
## StdDev: 10.47345 12.74987  
##   
## Fixed effects: Y ~ -1 + N + V   
## Value Std.Error DF t-value p-value  
## N0 73.04167 8.220351 51 8.885468 0.0000  
## N0.2 92.54167 8.220351 51 11.257630 0.0000  
## N0.4 107.87500 8.220351 51 13.122919 0.0000  
## N0.6 117.04167 8.220351 51 14.238038 0.0000  
## VGoldenRain 6.87500 7.078909 11 0.971195 0.3523  
## VMarvellous 12.16667 7.078909 11 1.718721 0.1136  
## Correlation:   
## N0 N0.2 N0.4 N0.6 VGldnR  
## N0.2 0.866   
## N0.4 0.866 0.866   
## N0.6 0.866 0.866 0.866   
## VGoldenRain -0.431 -0.431 -0.431 -0.431   
## VMarvellous -0.431 -0.431 -0.431 -0.431 0.500  
##   
## Standardized Within-Group Residuals:  
## Min Q1 Med Q3 Max   
## -1.84134149 -0.66279741 -0.06694262 0.63822477 1.66066788   
##   
## Number of Observations: 72  
## Number of Groups:   
## B V %in% B   
## 6 18

## Using lmer (lme4)  
m\_lmer <- lmer(Y ~ -1 + N + V + (1|B/V), data=data)  
summary(m\_lmer)

## Linear mixed model fit by REML ['lmerMod']  
## Formula: Y ~ -1 + N + V + (1 | B/V)  
## Data: data  
##   
## REML criterion at convergence: 568.1  
##   
## Scaled residuals:   
## Min 1Q Median 3Q Max   
## -1.84135 -0.66280 -0.06694 0.63822 1.66067   
##   
## Random effects:  
## Groups Name Variance Std.Dev.  
## V:B (Intercept) 109.7 10.47   
## B (Intercept) 214.5 14.65   
## Residual 162.6 12.75   
## Number of obs: 72, groups: V:B, 18; B, 6  
##   
## Fixed effects:  
## Estimate Std. Error t value  
## N0 73.042 8.220 8.885  
## N0.2 92.542 8.220 11.258  
## N0.4 107.875 8.220 13.123  
## N0.6 117.042 8.220 14.238  
## VGoldenRain 6.875 7.079 0.971  
## VMarvellous 12.167 7.079 1.719  
##   
## Correlation of Fixed Effects:  
## N0 N0.2 N0.4 N0.6 VGldnR  
## N0.2 0.866   
## N0.4 0.866 0.866   
## N0.6 0.866 0.866 0.866   
## VGoldenRain -0.431 -0.431 -0.431 -0.431   
## VMarvellous -0.431 -0.431 -0.431 -0.431 0.500

#### 4.

# BLUP estimates (for random terms block, variety within block)  
BLUP\_lme <- ranef(m\_lme)  
BLUP\_reg <- BLUP(m\_reg)  
  
# or with specific term targeted  
BLUP\_lme\_B <- ranef(m\_lme, level = "B")  
BLUP\_reg\_B <- BLUP(m\_reg, RE = "B")  
  
# compare  
BLUP\_lme\_vec <- c(unlist(BLUP\_lme$B), unlist(BLUP\_lme$V))  
BLUP\_reg\_vec <- BLUP\_reg$Mean  
  
BLUP\_comp <- cbind(BLUP\_lme\_vec, BLUP\_reg\_vec)  
  
# BLUE estimates variety  
m\_lme <- lme(Y~-1+V+N,random=~1|B/V,data=data,method="REML")  
BLUE\_V <- m\_lme$coefficients$fixed[1:3]  
  
m\_reg <- regress(Y~-1+V+N,~B+I(B:V),identity=TRUE,verbose=0,data=data)  
BLUE\_V <- m\_reg$beta[1:3, 1]  
  
# Correspond to the variety effect in the reference (0) fertilisation level.

#### 5.

# manual computation of the BLUP using MM formulas  
y <- data$Y  
N <- nrow(data)  
I <- diag(N)  
X <- model.matrix(~ -1 + N + V, data = data)  
ZB <- model.matrix(~ -1 + B , data = data)  
ZBV <- model.matrix(~ -1 + V:B , data = data)  
  
# Possibility to get the incidences matrix from  
# regress output  
ZBV\_reg <- m\_reg$Z$`I(B:V)`  
  
sB <- m\_reg$sigma[1]  
sBV <- m\_reg$sigma[2]  
sE <- m\_reg$sigma[3]  
  
lbd\_B <- sE/sB  
lbd\_BV <- sE/sBV  
  
XX <- crossprod(X, X)  
XZB <- t(X) %\*% ZB  
ZBX <- t(ZB) %\*% X  
XZBV <- t(X) %\*% ZBV  
ZBVX <- t(ZBV) %\*% X  
  
ZBZB <- crossprod(ZB, ZB)  
ZBZBV <- crossprod(ZB, ZBV)  
  
ZBVZBV <- crossprod(ZBV, ZBV)  
ZBVZB <- crossprod(ZBV, ZB)  
  
ZZ\_B <- ZBZB + (lbd\_B \* diag(nrow(ZBZB)))  
ZZ\_BV <- ZBVZBV + (lbd\_BV \* diag(nrow(ZBVZBV)))  
  
# MM equations  
C <- rbind(cbind(XX, XZB, XZBV),  
 cbind(ZBX, ZZ\_B, ZBZBV),  
 cbind(ZBVX, ZBVZB, ZZ\_BV))  
rhs <- rbind(t(X) %\*% y, t(ZB) %\*% y, t(ZBV) %\*% y)  
BU <- solve(C) %\*% rhs  
  
# compare with the estimates returned by regress  
BU\_reg <- c(m\_reg$beta, BLUP\_reg\_vec)  
BU\_comp <- data.frame(MM\_est = BU, reg\_est = BU\_reg)

#### Ex2 Phenotypic data analysis in BCNAM (Lata data)

This exercise use real data from the WCA-BCNAM Lata population. It contains a set of 1082 genotypes that were phenotyped in three environment according to an alpha design with two replicates (r=2).

1. Look at the data and try to grasp the elements of the design

load("./data/pheno\_BCNAM.RData")  
head(data)

## env rep block row col cross geno PH geno\_env  
## 1 KOL 1 1 1 1 BC58 BC58-1010 290.0 BC58-1010\_KOL  
## 2 KOL 1 1 2 1 BC50 BC50-0117 221.0 BC50-0117\_KOL  
## 3 KOL 1 1 3 1 BC61 BC61-0914 258.5 BC61-0914\_KOL  
## 4 KOL 1 1 4 1 BC55 BC55-0751 248.5 BC55-0751\_KOL  
## 5 KOL 1 1 5 1 BC52 BC52-0550 261.0 BC52-0550\_KOL  
## 6 KOL 1 1 6 1 BC53 BC53-0740 208.5 BC53-0740\_KOL

1. Think about a mixed model to analyse those data and calculate the heritability at the whole population level. At that stage, ignore the fact that the data are structured in crosses.
2. Calculate your model using the lmer() function from the lme4 package.
3. Obtain the variance component estimates to calculate the heritability using the standard formula

For that purpose you can use R code similar to this one

remat <- summary(m)  
Verr <- sigma(m)^2  
Verr <- remat$sigma^2  
V\_var <- remat$varcor  
Vg <- V\_var$geno[1, 1]  
Vge <- V\_var$geno\_env[1, 1]

1. Get the fixed effect (BLUE) and random effect (BLUP) estimates. For that you can use the following R code

BLUE <- m@beta  
BLUP <- getME(m, name = "b") # random effect from MM  
BLUP2 <- m@u # spherical random effect

1. “Manual” estimation of the BLUE and the BLUP using the mixed model equations. Try to understand the code below and make connection with the following expression (Attention: in the R code the terms are reversed). Compare the value you obtain with the ones provided by the function.

# Collect the design matrices  
X <- getME(m, name = "X")  
Ztlist <- getME(m, name = "Ztlist")  
  
D <- vector(mode = "list", length = length(Ztlist) + 1)  
names(D) <- c("Xenv","ZGxE", "ZG", "ZBl", "ZRep")  
D[[1]] <- t(X)  
for(i in 2:length(D)){D[[i]] <- Ztlist[[i-1]]}  
  
# Collect the variance components  
Ve <- sigma(m)^2  
V\_var <- summary(m)$varcor  
Vr <- V\_var$`env:rep`[1, 1]  
Vb <- V\_var$`env:rep:block`[1, 1]  
Vg <- V\_var$geno[1, 1]  
Vge <- V\_var$geno\_env[1, 1]  
  
# Collect the parameter name  
p\_names <- c()  
for(i in 1:length(D)){  
 p\_names <- c(p\_names, paste0(names(D)[i], "\_",rownames(D[[i]])))  
}  
  
# form all the cross products  
crpod\_list <- vector(mode = "list", length = length(D))  
for(i in 1:length(crpod\_list)){  
 crpod\_list[[i]] <- vector(mode = "list", length = length(D))  
}  
  
for(i in 1:length(D)){  
 for(j in 1:length(D)){  
 crpod\_list[[i]][[j]] <- D[[i]] %\*% t(D[[j]])  
 }  
}  
  
# Add the diagonal elements  
V\_vec <- c(NA, Vge, Vg, Vb, Vr)  
for(i in 2:length(D)){  
 CP <- crpod\_list[[i]][[i]]  
 crpod\_list[[i]][[i]] <- CP + (diag(nrow(CP)) \* (Ve/V\_vec[i]))  
}  
  
# Form the C or lhs matrix or lhs  
C <- rbind(Reduce(f = cbind, x = crpod\_list[[1]]),  
 Reduce(f = cbind, x = crpod\_list[[2]]),  
 Reduce(f = cbind, x = crpod\_list[[3]]),  
 Reduce(f = cbind, x = crpod\_list[[4]]),  
 Reduce(f = cbind, x = crpod\_list[[5]]))  
  
# calculate the inverse  
C\_i <- solve(C)  
  
# compose the rhs matrix (get the data used by the model)  
y <- getME(m, name = "y")  
RHS <- c()  
for(i in 1:length(D)){  
 Xt <- D[[i]]  
 RHS\_i <- D[[i]] %\*% y  
 RHS <- rbind(RHS, RHS\_i)  
}  
  
BU <- matrix(C\_i %\*% RHS, ncol = 1)  
BU <- data.frame(p\_id = p\_names, est = BU)

1. Look at the distribution of the genotype BLUP using boxplot. Why are they centered at 0?
2. Try to fit the same model as before using the regress package. For that purpose you can use the following code. What do you notice in terms of estimation time.

m\_reg <- regress(formula = PH ~ -1 + env, Vformula = ~ I(env:rep) + I(env:rep:block) +  
 (geno) + (geno\_env), data=data)  
# summary(m\_reg)

1. The data are actually structured in crosses. Look at he following R code. Try to understand what are the changes and execute.

# use nlme (lme fct): more complex VCOV structure but  
# random effect are nested  
  
# pass the env:rep:block in the fixed part  
m <- lme(PH ~ env:rep:block + cross,   
 random = list(geno = pdDiag(form = ~ -1 + cross),  
 geno\_env = pdIdent(form = ~ 1)),  
 control = list(opt = "optim", maxIter = 100, msMaxIter = 100,  
 msVerbose = TRUE),  
 weights = varIdent(form = ~ 1 | cross),  
 data=data, na.action = na.omit)  
  
# simplify the model  
m <- lme(PH ~ env:rep + cross,   
 random = list(geno = pdDiag(form = ~ -1 + cross),  
 geno\_env = pdIdent(form = ~ 1)),  
 control = list(opt = "optim", maxIter = 100, msMaxIter = 100,  
 msVerbose = TRUE),  
 weights = varIdent(form = ~ 1 | cross),  
 data=data, na.action = na.omit)  
  
# simplify again the model  
m <- lme(PH ~ env + cross,   
 random = list(geno = pdDiag(form = ~ -1 + cross),  
 geno\_env = pdIdent(form = ~ 1)),  
 control = list(opt = "optim", maxIter = 100, msMaxIter = 100,  
 msVerbose = TRUE),  
 weights = varIdent(form = ~ 1 | cross),  
 data=data, na.action = na.omit)  
  
# check the output  
summary(m)  
  
# get the estimates  
BU <- m$coefficients  
B <- BU$fixed  
U <- BU$random  
# Normally genotype BLUP  
UG <- U$geno

1. Using the following code, collect the new variance estimates. Try to connect that information to the model specification and to the model output (summary())

# cross-specific error terms  
Ve <- (c(1.0000000, coef(m$modelStruct$varStruct, unconstrained=F))\*m$sigma)^2  
Ve <- Ve[-length(Ve)]  
  
# within cross genotypic variance  
vc <- VarCorr(m)  
suppressWarnings(storage.mode(vc) <- "numeric")  
  
n\_cr <- length(Ve)  
  
Vg <- vc[2:(n\_cr + 1), 1]  
Vge <- vc[(n\_cr + 4), 1]

1. Using the cross-specific varianc estimates re-calculate the heritability and compare with the previous global estimate.
2. Let assume that we would like to obtained genotype BLUEs, modify the previous model and re-estimate it to get the BLUEs.

#### Answer

#### 2.

We could assume the following mixed model

#### 3.

m <- lmer(PH ~ -1 + env + (1|env:rep) + (1|env:rep:block) + (1|geno) + (1|geno\_env), data=data)  
# summary(m)  
  
# alternative (environment as random)  
# m <- lmer(PH ~ -1 + (1|env) + (1|env:rep) + (1|env:rep:block) + (1|geno) + (1|geno\_env), data=data)  
# summary(m)

#### 4.

remat <- summary(m)  
Verr <- sigma(m)^2  
Verr <- remat$sigma^2  
V\_var <- remat$varcor  
Vg <- V\_var$geno[1, 1]  
Vge <- V\_var$geno\_env[1, 1]  
  
n\_env <- 3  
n\_rep = 2  
h2 <- (Vg)/(Vg + (Vge/n\_env) + (Verr/(n\_env\*n\_rep)))

#### 5.

BLUE <- m@beta  
BLUP <- getME(m, name = "b") # random effect from MM  
BLUP2 <- m@u # spherical random effect

#### 6.

# Collect the design matrices  
X <- getME(m, name = "X")  
Ztlist <- getME(m, name = "Ztlist")  
  
D <- vector(mode = "list", length = length(Ztlist) + 1)  
names(D) <- c("Xenv","ZGxE", "ZG", "ZBl", "ZRep")  
D[[1]] <- t(X)  
for(i in 2:length(D)){D[[i]] <- Ztlist[[i-1]]}  
  
# Collect the variance components  
Ve <- sigma(m)^2  
V\_var <- summary(m)$varcor  
Vr <- V\_var$`env:rep`[1, 1]  
Vb <- V\_var$`env:rep:block`[1, 1]  
Vg <- V\_var$geno[1, 1]  
Vge <- V\_var$geno\_env[1, 1]  
  
# Collect the parameter name  
p\_names <- c()  
for(i in 1:length(D)){  
 p\_names <- c(p\_names, paste0(names(D)[i], "\_",rownames(D[[i]])))  
}  
  
# form all the cross products  
crpod\_list <- vector(mode = "list", length = length(D))  
for(i in 1:length(crpod\_list)){  
 crpod\_list[[i]] <- vector(mode = "list", length = length(D))  
}  
  
for(i in 1:length(D)){  
 for(j in 1:length(D)){  
 crpod\_list[[i]][[j]] <- D[[i]] %\*% t(D[[j]])  
 }  
}  
  
# Add the diagonal elements  
V\_vec <- c(NA, Vge, Vg, Vb, Vr)  
for(i in 2:length(D)){  
 CP <- crpod\_list[[i]][[i]]  
 crpod\_list[[i]][[i]] <- CP + (diag(nrow(CP)) \* (Ve/V\_vec[i]))  
}  
  
# Form the C or lhs matrix or lhs  
C <- rbind(Reduce(f = cbind, x = crpod\_list[[1]]),  
 Reduce(f = cbind, x = crpod\_list[[2]]),  
 Reduce(f = cbind, x = crpod\_list[[3]]),  
 Reduce(f = cbind, x = crpod\_list[[4]]),  
 Reduce(f = cbind, x = crpod\_list[[5]]))  
  
# calculate the inverse  
C\_i <- solve(C)  
  
# compose the rhs matrix (get the data used by the model)  
y <- getME(m, name = "y")  
RHS <- c()  
for(i in 1:length(D)){  
 Xt <- D[[i]]  
 RHS\_i <- D[[i]] %\*% y  
 RHS <- rbind(RHS, RHS\_i)  
}  
  
BU <- matrix(C\_i %\*% RHS, ncol = 1)  
BU <- data.frame(p\_id = p\_names, est = BU)  
  
# compare  
BLUE - BU[1:3, 2]

## [1] 1.220798e-09 1.671935e-09 1.771809e-09

sum(round(BLUP2 - BU[4:nrow(BU), 2], 10))

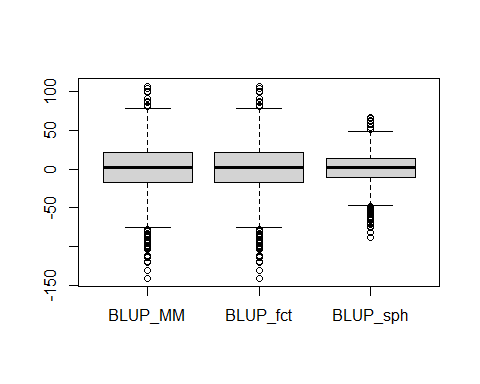
## [1] -6.57e-08

d\_BLUP <- data.frame(p\_id = p\_names[4:nrow(BU)],   
 BLUP\_MM = BU[4:nrow(BU), 2],  
 BLUP\_fct = matrix(getME(m, "b"), ncol = 1),  
 BLUP\_sph = m@u)  
  
# extract the BLUP associated to a single random term (genotype)  
d\_BLUP\_G <- d\_BLUP[grep(pattern = "ZG\_", x = d\_BLUP$p\_id), ]

#### 7.

Genotype BLUP distribution

# extract the BLUP associated to a single random term (genotype)  
d\_BLUP\_G <- d\_BLUP[grep(pattern = "ZG\_", x = d\_BLUP$p\_id), ]  
  
boxplot(d\_BLUP\_G[, 2:4])



The BLUP are centred at zero because we assume that they come from a distribution . Therefore, there expetation is equalt to .

#### 8.

the regress() function takes more time to converge.

#### 9.

Heritability model computation with the addition of the cross term.

# use nlme (lme fct): more complex VCOV structure but  
# random effect are nested  
  
# pass the env:rep:block in the fixed part  
# m <- lme(PH ~ env:rep:block + cross,   
# random = list(geno = pdDiag(form = ~ -1 + cross),  
# geno\_env = pdIdent(form = ~ 1)),  
# control = list(opt = "optim", maxIter = 100, msMaxIter = 100,  
# msVerbose = TRUE),  
# weights = varIdent(form = ~ 1 | cross),  
# data=data, na.action = na.omit)  
  
# simplify the model  
# m <- lme(PH ~ env:rep + cross,   
# random = list(geno = pdDiag(form = ~ -1 + cross),  
# geno\_env = pdIdent(form = ~ 1)),  
# control = list(opt = "optim", maxIter = 100, msMaxIter = 100,  
# msVerbose = TRUE),  
# weights = varIdent(form = ~ 1 | cross),  
# data=data, na.action = na.omit)  
  
# simplify again the model  
m <- lme(PH ~ env + cross,   
 random = list(geno = pdDiag(form = ~ -1 + cross),  
 geno\_env = pdIdent(form = ~ 1)),  
 control = list(opt = "optim", maxIter = 100, msMaxIter = 100,  
 msVerbose = TRUE),  
 weights = varIdent(form = ~ 1 | cross),  
 data=data, na.action = na.omit)

## initial value 51132.922648   
## iter 10 value 50899.731657  
## iter 20 value 50879.674684  
## iter 30 value 50875.653438  
## final value 50875.456595   
## converged

# check the output  
summary(m)

## Linear mixed-effects model fit by REML  
## Data: data   
## AIC BIC logLik  
## 63373.61 63658.2 -31644.8  
##   
## Random effects:  
## Formula: ~-1 + cross | geno  
## Structure: Diagonal  
## crossBC50 crossBC51 crossBC52 crossBC53 crossBC54 crossBC55 crossBC56  
## StdDev: 39.05246 58.3868 35.32785 22.5223 23.46637 16.53119 30.73826  
## crossBC57 crossBC58 crossBC59 crossBC60 crossBC61 crossBC62  
## StdDev: 32.782 32.19834 19.76203 28.33136 25.5591 37.93671  
##   
## Formula: ~1 | geno\_env %in% geno  
## (Intercept) Residual  
## StdDev: 2.926925 26.96079  
##   
## Variance function:  
## Structure: Different standard deviations per stratum  
## Formula: ~1 | cross   
## Parameter estimates:  
## BC50 BC51 BC52 BC53 BC54 BC55 BC56 BC57   
## 1.0000000 1.0330374 0.9898153 0.9464426 1.0581503 0.9436099 0.9513678 0.8858233   
## BC58 BC59 BC60 BC61 BC62   
## 1.2058249 0.9100397 1.0011972 0.9350923 0.9822198   
## Fixed effects: PH ~ env + cross   
## Value Std.Error DF t-value p-value  
## (Intercept) 214.50355 4.087940 3246 52.47229 0.0000  
## envSAM\_HP 53.85892 0.812919 2162 66.25375 0.0000  
## envSAM\_LP 18.30542 0.812878 2162 22.51926 0.0000  
## crossBC51 9.65212 8.747034 1069 1.10347 0.2701  
## crossBC52 9.54688 5.797502 1069 1.64672 0.0999  
## crossBC53 36.28775 4.875341 1069 7.44312 0.0000  
## crossBC54 8.07970 5.011373 1069 1.61227 0.1072  
## crossBC55 30.48333 4.687938 1069 6.50250 0.0000  
## crossBC56 37.76262 5.192278 1069 7.27284 0.0000  
## crossBC57 -1.79580 5.581587 1069 -0.32174 0.7477  
## crossBC58 56.86889 5.478080 1069 10.38117 0.0000  
## crossBC59 35.85185 4.767778 1069 7.51961 0.0000  
## crossBC60 34.49375 5.298864 1069 6.50965 0.0000  
## crossBC61 26.24694 4.926158 1069 5.32807 0.0000  
## crossBC62 14.43854 5.998130 1069 2.40717 0.0162  
## Correlation:   
## (Intr) eSAM\_H eSAM\_L crBC51 crBC52 crBC53 crBC54 crBC55 crBC56 crBC57  
## envSAM\_HP -0.099   
## envSAM\_LP -0.099 0.500   
## crossBC51 -0.461 0.000 0.000   
## crossBC52 -0.696 0.000 0.000 0.325   
## crossBC53 -0.827 0.000 0.000 0.387 0.583   
## crossBC54 -0.805 0.000 0.000 0.376 0.568 0.675   
## crossBC55 -0.861 0.000 0.000 0.402 0.607 0.722 0.702   
## crossBC56 -0.777 0.000 0.000 0.363 0.548 0.651 0.634 0.677   
## crossBC57 -0.723 0.000 0.000 0.338 0.510 0.606 0.590 0.630 0.569   
## crossBC58 -0.736 0.000 0.000 0.344 0.519 0.617 0.601 0.642 0.580 0.539  
## crossBC59 -0.846 0.000 0.000 0.395 0.597 0.709 0.690 0.738 0.666 0.620  
## crossBC60 -0.761 0.000 0.000 0.356 0.537 0.638 0.621 0.664 0.599 0.558  
## crossBC61 -0.819 0.000 0.000 0.383 0.577 0.687 0.668 0.714 0.645 0.600  
## crossBC62 -0.673 0.000 0.000 0.314 0.474 0.564 0.549 0.586 0.530 0.493  
## crBC58 crBC59 crBC60 crBC61  
## envSAM\_HP   
## envSAM\_LP   
## crossBC51   
## crossBC52   
## crossBC53   
## crossBC54   
## crossBC55   
## crossBC56   
## crossBC57   
## crossBC58   
## crossBC59 0.631   
## crossBC60 0.568 0.653   
## crossBC61 0.611 0.702 0.632   
## crossBC62 0.502 0.577 0.519 0.558  
##   
## Standardized Within-Group Residuals:  
## Min Q1 Med Q3 Max   
## -4.72461929 -0.58008692 -0.01240365 0.59350403 4.45653523   
##   
## Number of Observations: 6492  
## Number of Groups:   
## geno geno\_env %in% geno   
## 1082 3246

# get the estimates  
BU <- m$coefficients  
B <- BU$fixed  
U <- BU$random  
# Normally genotype BLUP  
UG <- U$geno

#### 10.

We can notice that the genotype variance estimates as well as the error estimates are now cross specific.

# cross-specific error terms  
Ve <- (c(1.0000000, coef(m$modelStruct$varStruct, unconstrained=F))\*m$sigma)^2  
Ve <- Ve[-length(Ve)]  
  
# within cross genotypic variance  
vc <- VarCorr(m)  
suppressWarnings(storage.mode(vc) <- "numeric")  
  
n\_cr <- length(Ve)  
  
Vg <- vc[2:(n\_cr + 1), 1]  
Vge <- vc[(n\_cr + 4), 1]  
  
Ve  
Vg

#### 11.

# heritability computation  
n\_env <- 3  
n\_rep <- 2  
h2\_cr <- Vg / (Vg + (Vge/n\_env) + (Ve/(n\_env \* n\_rep)))  
  
res <- data.frame(Vg, Vge, Ve, h2\_cr)

We notice that compare to the general estimate there is quite some variability in terms of heritability between the different crosses.

#### 12.

m <- lmer(PH ~ -1 + env + geno + (1|env:rep) + (1|env:rep:block) + (1|geno\_env), data=data)  
summary(m)  
  
# get the (genotype) BLUE  
m\_sum <- summary(m)  
coeff <- m\_sum$coefficients  
BLUE <- coeff[, 1]  
names(BLUE) <- rownames(coeff)  
  
# remove the geno in the BLUE name  
names(BLUE) <- gsub(pattern = "geno", replacement = "", x = names(BLUE))  
  
# here we set the reference genotype as the average score in the three environment  
# The other BLUEs are defined as a deviation with respect to this reference value.  
G\_ref <- mean(BLUE[1:3])  
BLUE\_geno <- BLUE[4:length(BLUE)]  
BLUE\_geno <- c(G\_ref, BLUE\_geno + G\_ref)  
  
  
# find the reference (missing genotype)  
geno\_id <- unique(as.character(data$geno))  
geno\_m <- geno\_id[!(geno\_id %in% names(BLUE\_geno))]  
geno\_m <- geno\_m[!is.na(geno\_m)]  
names(BLUE\_geno)[1] <- geno\_m  
  
plot(x = d\_BLUP\_G$BLUP\_MM, y = BLUE\_geno)  
cor(x = d\_BLUP\_G$BLUP\_MM, y = BLUE\_geno)  
  
# alternative set the genotype as the first term without intercept  
m2 <- lmer(PH ~ -1 + geno + env + (1|env:rep) + (1|env:rep:block) + (1|geno\_env), data=data)  
summary(m2)  
  
# get the (genotype) BLUE  
m\_sum2 <- summary(m2)  
coeff2 <- m\_sum2$coefficients  
BLUE2 <- coeff2[, 1]  
names(BLUE2) <- rownames(coeff2)  
  
BLUE\_geno2 <- BLUE2[1:1082]  
BLUE\_geno2[1000:1082]  
  
cor(BLUE\_geno, BLUE\_geno2)

#### Genetic model (Animal, GBLUP)

1. Using the following data Calculate a kinship matrix using the A.mat function from rrBLUP. Do not forget to add a small digit to the diagonal of the kinship matrix to facilitate its inversion.

library(rrBLUP)  
library(sommer)

## Le chargement a nécessité le package : MASS

##   
## Attachement du package : 'MASS'

## L'objet suivant est masqué depuis 'package:dplyr':  
##   
## select

## Le chargement a nécessité le package : crayon

##   
## Attachement du package : 'crayon'

## L'objet suivant est masqué depuis 'package:ggplot2':  
##   
## %+%

##   
## Attachement du package : 'sommer'

## Les objets suivants sont masqués depuis 'package:rrBLUP':  
##   
## A.mat, GWAS

y <- c(0.19, 1.23, 0.86, 1.23, 0.45)  
x <- matrix(c(0, 1, 1, 1, 0,  
 0, 0, 0, 1, 1,  
 0, 0, 0, 1, 1,  
 0, 1, 1, 1, 1,  
 0, 1, 0, 0, 1,  
 0, 1, 0, 1, 1,  
 1, 2, 1, 2, 2,  
 2, 1, 1, 1, 1,  
 0, 0, 1, 1, 0,  
 2, 1, 1, 1, 1), nrow = 5, ncol = 10)  
GID <- paste0("G", 1:length(y))  
d <- data.frame(GID, y)

1. get the error and genetic variance estimates using the mmer function from sommer
2. using the VCOV estimates build the MM equation to estimate the BLUE and BLUP. Compare those estimates with the one returned by sommer. You can get those estimate using

BLUP <- m$U$`u:GID`$y

1. Estimation of the breeding values (BLUP) with missing values. A convenient way to predict breeding values is to set those values as missing and estimate the model as usual. You will get an estimate for those values. This is what is done is the piece of code below. Try to understand what happens by reading the code, especially in terms of relatedness between the genotype with missing phenotype and the rest of the population. What is the implication of that in term of EBV estimation?

# Get data  
load(file = "./data/A\_mat.RData")  
y <- c(354, 251, 327, 328, 301, 270, 330, NA, NA)  
N <- length(y)  
GID <- paste0("G", 1:length(y))  
A <- cbind(A, matrix(0, nrow = 7, ncol = 2))  
A <- rbind(A, matrix(0, nrow = 2, ncol = 9))  
rownames(A) <- colnames(A) <- GID  
  
A[8, 8] <- A[9, 9] <- 1 # identity for same individual  
  
d <- data.frame(GID, y)  
Ai <- as(solve(A), Class = "dgCMatrix")  
m <- mmec(y ~ 1,  
 random= ~vsc(isc(GID),Gu=Ai),  
 rcov= ~ units,  
 data=d, verbose = FALSE, dateWarning = FALSE)

## Adding additional levels of Gu in the model matrix of 'GID'

sG <- m$sigma[1] # genetic variance  
sE <- m$sigma[2] # error (environment) variance  
  
BLUP <- m$u  
BLUP

## [,1]  
## G1 45.892774  
## G2 -56.906514  
## G3 19.006453  
## G4 20.003812  
## G5 -6.998058  
## G6 -37.978752  
## G7 21.952409  
## G8 0.000000  
## G9 0.000000

1. (Optional) You can try to build the MM equation for the previous situation. Otherwise, you can have a look to the code below

# manual prediction  
y <- y[!is.na(y)] # need to remove the missing values  
N <- length(y)  
X <- matrix(rep(1, N), nrow = N)  
  
# the obtention of values for the genotype with missing information  
# is realized through the Z matrix. In that matrix, unobserved individuals  
# receive a column with only 0 values. Therefore, the dimension of the Z  
# matrix is [N\_obs x N\_geno]  
Z <- cbind(diag(N), rep(0, N), rep(0, N))  
colnames(Z) <- GID  
  
B\_hat <- m$b  
lambda <- sE/sG  
  
XX <- crossprod(X, X)  
XZ <- t(X) %\*% Z  
ZX <- t(Z) %\*% X  
ZZ <- crossprod(Z, Z) + lambda \* solve(A)  
  
# MM equations  
C <- rbind(cbind(XX, XZ), cbind(ZX, ZZ))  
rhs <- rbind(t(X) %\*% y, t(Z) %\*% y)  
  
# solutions  
BU <- solve(C) %\*% rhs  
  
# Compare  
d\_BLUP <- data.frame(BLUP\_som = BLUP, BLUP\_man = BU[2:length(BU)])

1. Let us now add some relatedness between the genotype with missing phenotype record and the rest of the population. Explain what does the modification below means in terms of coancestry or pedigree relatedness.

A[8, 1] <- A[1, 8] <- 0.5

1. Let us now recalculate the breeding values. What do you observe? How can you explain that?

Ai <- as(solve(A), Class = "dgCMatrix")  
m <- mmec(y ~ 1,  
 random= ~vsc(isc(GID),Gu=Ai),  
 rcov= ~ units,  
 data=d, verbose = FALSE, dateWarning = FALSE)

## Adding additional levels of Gu in the model matrix of 'GID'

BLUP <- m$u  
  
# MM solve  
ZZ <- crossprod(Z, Z) + lambda \* solve(A)  
  
# MM equations  
C <- rbind(cbind(XX, XZ), cbind(ZX, ZZ))  
rhs <- rbind(t(X) %\*% y, t(Z) %\*% y)  
  
# solutions  
BU <- solve(C) %\*% rhs  
  
# Compare  
d\_BLUP <- data.frame(BLUP\_som = BLUP, BLUP\_man = BU[2:length(BU)])  
d\_BLUP

## BLUP\_som BLUP\_man  
## G1 45.892774 45.90621  
## G2 -56.906514 -56.91825  
## G3 19.006453 19.00566  
## G4 20.003812 20.00335  
## G5 -6.998058 -6.99830  
## G6 -37.978752 -37.98143  
## G7 21.952409 21.95838  
## G8 25.986151 25.99831  
## G9 0.000000 0.00000

1. EBV interpretation.

8.a) Follow the development bellow.

Let us remember that the general expression for the BLUP in a mixed model is the following

Using the following equality (for a proof see Sathoh 2018 An alternative derivation method of mixed model equations …)

We can rewrite the BLUP expression as such

In the animal model and Therefore,

Considering that we can multiply the inside terms of the left parenthesis by (to be checked), we obtain

We can further reduce reformulate the expression by substituting

In the case of a single observation (each genotype is observed one time) , so . We can also consider that is the deviation of the phenotype observation with respect to the central tendency and call it . Then

8.b) Try to interpret how the BLUP () or the expected breeding value are constructed using the following piece of code

# BLUP calculation via the "direct" formula  
u\_bar <- solve(crossprod(Z, Z) + lambda \* solve(A)) %\*% t(Z) %\*% (y - (X %\*% B\_hat))  
d\_comp <- data.frame(som\_BLUP = BLUP, MM\_sol = BU[-1, ], u\_bar)  
d\_comp

## som\_BLUP MM\_sol u\_bar  
## G1 45.892774 45.90621 45.905711  
## G2 -56.906514 -56.91825 -56.918744  
## G3 19.006453 19.00566 19.005160  
## G4 20.003812 20.00335 20.002849  
## G5 -6.998058 -6.99830 -6.998799  
## G6 -37.978752 -37.98143 -37.981931  
## G7 21.952409 21.95838 21.957882  
## G8 25.986151 25.99831 25.998231  
## G9 0.000000 0.00000 0.000000

# decomposition of the formula  
y\_adj <- t(Z) %\*% (y - (X %\*% B\_hat))  
W\_G <- solve(crossprod(Z, Z) + lambda \* solve(A))  
  
# shape of W\_G "projection" matrix  
round(W\_G, 6)

## G1 G2 G3 G4 G5 G6 G7  
## G1 0.996257 -0.000859 0.001151 0.001151 0.001723 -0.000001 0.000000  
## G2 -0.000859 0.996831 -0.000001 -0.000001 0.001724 0.001152 0.000000  
## G3 0.001151 -0.000001 0.997691 0.000001 0.000002 0.000000 0.000000  
## G4 0.001151 -0.000001 0.000001 0.997691 0.000002 0.000000 0.000000  
## G5 0.001723 0.001724 0.000002 0.000002 0.996544 0.000002 0.000000  
## G6 -0.000001 0.001152 0.000000 0.000000 0.000002 0.997691 0.000000  
## G7 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.998266  
## G8 1.077435 0.247415 -0.331319 -0.331319 -0.495976 0.000286 0.000000  
## G9 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000  
## G8 G9  
## G1 1.077435 0.0000  
## G2 0.247415 0.0000  
## G3 -0.331319 0.0000  
## G4 -0.331319 0.0000  
## G5 -0.495976 0.0000  
## G6 0.000286 0.0000  
## G7 0.000000 0.0000  
## G8 265.595053 0.0000  
## G9 0.000000 575.7755

# if h2 -> 1 ; lambda -> 0: the BLUP construction is mostly (only) based on  
# the direct observation and not the "borrowed" information via kinship relatedness.  
  
# fist individual  
W\_G1 <- round(W\_G[1, ], 3)  
W\_G1

## G1 G2 G3 G4 G5 G6 G7 G8 G9   
## 0.996 -0.001 0.001 0.001 0.002 0.000 0.000 1.077 0.000

u\_1 <- W\_G[1, ] %\*% y\_adj  
u\_1

## [,1]  
## [1,] 45.90571

BLUP[1]

## [1] 45.89277

# for the 8th individual there is no direct observation to it is a weighted sum  
# of the other individual deviation proportional to the relatedness given by  
# the relationship matrix  
  
# 8th individual  
W\_G8 <- round(W\_G[8, ], 3)  
W\_G8

## G1 G2 G3 G4 G5 G6 G7 G8 G9   
## 1.077 0.247 -0.331 -0.331 -0.496 0.000 0.000 265.595 0.000

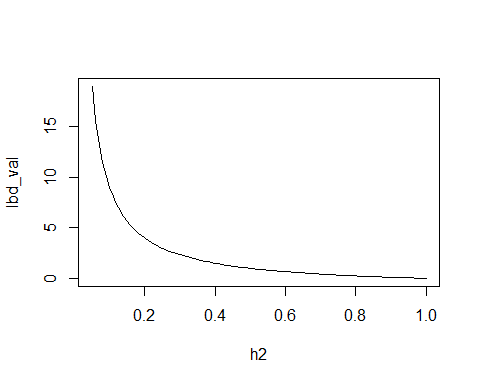
u\_8 <- W\_G[8, ] %\*% y\_adj  
u\_8

## [,1]  
## [1,] 25.99823

BLUP[8]

## [1] 25.98615

# if h2 -> 1 ; lambda -> Inf;  
  
lbd\_fct <- function(h2){(1-h2)/h2}  
h2 <- rev(seq(0.05, 1, by = 0.01))  
lbd\_val <- lbd\_fct(h2)  
plot(y = lbd\_val, x = h2, type = "l")



# you progressively increase the information coming from the  
# lbd \* A\_inv term. So you balance the direct observation by  
# undirect information from other observed individuals inversely  
# proportional to the lambda term.  
  
W\_G <- solve(crossprod(Z, Z) + 0.5 \* solve(A))  
  
# shape of W\_G "projection" matrix  
round(W\_G, 4)

## G1 G2 G3 G4 G5 G6 G7 G8 G9  
## G1 0.5507 -0.0399 0.1101 0.1101 0.1277 -0.0080 0.0000 0.4493 0  
## G2 -0.0399 0.5826 -0.0080 -0.0080 0.1357 0.1165 0.0000 0.0399 0  
## G3 0.1101 -0.0080 0.6220 0.0220 0.0255 -0.0016 0.0000 -0.1101 0  
## G4 0.1101 -0.0080 0.0220 0.6220 0.0255 -0.0016 0.0000 -0.1101 0  
## G5 0.1277 0.1357 0.0255 0.0255 0.5658 0.0271 0.0000 -0.1277 0  
## G6 -0.0080 0.1165 -0.0016 -0.0016 0.0271 0.6233 0.0000 0.0080 0  
## G7 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.6667 0.0000 0  
## G8 0.4493 0.0399 -0.1101 -0.1101 -0.1277 0.0080 0.0000 1.5507 0  
## G9 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 2

# fist individual  
W\_G1 <- round(W\_G[1, ], 3)  
W\_G1

## G1 G2 G3 G4 G5 G6 G7 G8 G9   
## 0.551 -0.040 0.110 0.110 0.128 -0.008 0.000 0.449 0.000

u\_1 <- W\_G[1, ] %\*% y\_adj  
u\_1

## [,1]  
## [1,] 31.30707

BLUP[1]

## [1] 45.89277

# 8th individual  
W\_G8 <- round(W\_G[8, ], 3)  
W\_G8

## G1 G2 G3 G4 G5 G6 G7 G8 G9   
## 0.449 0.040 -0.110 -0.110 -0.128 0.008 0.000 1.551 0.000

u\_8 <- W\_G[8, ] %\*% y\_adj  
u\_8

## [,1]  
## [1,] 14.68895

BLUP[8]

## [1] 25.98615

# For the genotypes that are observed, the BLUP construction  
# first combine some information form the direct observation  
# and other relative is the heritability tend to 0.  
  
# For the genotypes that are not observed, the BLUP use the  
# relative observation proportional to the relatedness and  
# heritability.

#### Answer

#### 1.

y <- c(0.19, 1.23, 0.86, 1.23, 0.45)  
x <- matrix(c(0, 1, 1, 1, 0,  
 0, 0, 0, 1, 1,  
 0, 0, 0, 1, 1,  
 0, 1, 1, 1, 1,  
 0, 1, 0, 0, 1,  
 0, 1, 0, 1, 1,  
 1, 2, 1, 2, 2,  
 2, 1, 1, 1, 1,  
 0, 0, 1, 1, 0,  
 2, 1, 1, 1, 1), nrow = 5, ncol = 10)  
GID <- paste0("G", 1:length(y))  
d <- data.frame(GID, y)  
  
A <- A.mat(X = x -1)  
A <- A + diag(nrow(A))\*10^-6

#### 2.

rownames(A) <- colnames(A) <- GID  
m <- mmer(y ~ 1,  
 random= ~vsr(GID,Gu=A),  
 rcov= ~ units,  
 data=d, verbose = FALSE, dateWarning = FALSE)  
  
sA <- m$sigma[1]$`u:GID`[1, 1] # Additive genetic variance  
sE <- m$sigma[2]$units[1, 1] # Error variance

#### 3.

N <- length(y)  
X <- matrix(rep(1, N), nrow = N)  
Z <- diag(N)  
B\_hat <- m$Beta$Estimate  
lambda <- sE/sA  
  
XX <- crossprod(X, X)  
XZ <- t(X) %\*% Z  
ZX <- t(Z) %\*% X  
ZZ <- crossprod(Z, Z) + lambda \* solve(A)  
  
# MM equations  
C <- rbind(cbind(XX, XZ), cbind(ZX, ZZ))  
rhs <- rbind(t(X) %\*% y, t(Z) %\*% y)  
  
# solutions  
BU <- solve(C) %\*% rhs  
u\_bar <- solve(crossprod(Z, Z) + lambda \* solve(A)) %\*% t(Z) %\*% (y - X %\*% B\_hat)  
  
# compare with sommer  
data.frame(sommer = c(B\_hat, m$U$`u:GID`$y), MM\_solve = BU,  
 BLUP\_exp = c(B\_hat, u\_bar))

## sommer MM\_solve BLUP\_exp  
## 0.79200000 0.79200000 0.79200000  
## G1 -0.36720977 -0.36720977 -0.36720977  
## G2 0.14589068 0.14589068 0.14589068  
## G3 0.07512455 0.07512455 0.07512455  
## G4 0.18983726 0.18983726 0.18983726  
## G5 -0.04364272 -0.04364272 -0.04364272

#### 4.

The two genotypes with missing phenotype are considered to be unrelated with the rest of the population.

Since those individual do not have any phenotype record nor relatedness with the rest of the population, it is not possible to estimate their breeding value. It stays 0.

#### 6.

This means that genotype 1 and 8 are considered to be full-sibs

#### 7.

We can see that now genotype 8 get some estimate for its breeding value because of the relatedness with the genotype 1 that connect it to the rest of the population.

#### 8.

See formula and comments in the code.

#### Ex2 calculate the BLUP of a multiple random term model (A, G)

#### Ex1 Estimation of the prediction with NA values

### Rank deficiency method

#### Ex1: Interpretation of the shrinkage in Ridge regression

1. Using the singular value decomposition of , show that . For that use the fact that and are orthonormal matrix which means that and
2. Using and , show that the prediction from a Ridge regression can be expressed as
3. Since with representing the eigenvalues of how can we interprete the shrinkage applied to to obtain the predictions?

### Answer

#### 1.

#### 2.

#### 3.

The observed data are shrinked proportionally to . Since is the variance of the principal component. The shrinkage is proportional to the variability present in the predictor matrix.

#### Ex2: Determination of using cross validation

A standard way to estimate the shinkage parameter of the Ridge regression is to use cross-validation to determine the value of parameter that minimizes a certain loss function.

The principle of cross-validation is to separate part of the dataset to form a (pseudo) independent dataset (validation set), estimate the parameter on the rest of the data (training set), an evaluate the prediction ability of the function given estimated parameter on the validation set.

Here we would like to minimize the expected cross-validation error

where,

is the partition of the observation

is the estimate obtained on the training set (all observation minus the partition)

Based on this we can list the ingredients we need:

1. function to estimate
2. Way to predict new values:
3. Loss function
4. CV partition and loop

#### 1.

Using the small example data below, build the following mixed model equations and solve them using

y <- c(0.19, 1.23, 0.86, 1.23, 0.45)  
  
N <- length(y)  
  
X <- matrix(c(0,0,0,0,0,0,1,2,0,2,  
 1,0,0,1,1,1,2,1,0,1,  
 1,0,0,1,0,0,1,1,1,1,  
 1,1,1,1,0,1,2,1,1,1,  
 0,1,1,1,1,1,2,1,0,1), nrow = 5, byrow = TRUE)  
  
# identity matrix with dimension equal to the number of markers  
I <- diag(ncol(X))  
lambda <- 1  
M <- matrix(1, nrow = N, ncol = 1)

#### 2.

Convert your Ridge regression mixed model equation solver into a function that take among other the value of as a parameter. Test it with the example data on the following range of values for

lbd\_range <- seq(0.05, 2, 0.05)

#### 3.

Using the following cross-validation architecture fill the gaps and estimate the

for(i in 1:length(lbd\_range)){  
 for(j in 1:5){  
   
 # define TS and VS  
 M\_TS <- M[-j, , drop = FALSE]  
 X\_TS <- X[-j, , drop = FALSE]  
 y\_TS <- y[-j]  
   
 X\_VS <- cbind(1, X[j, , drop = FALSE]) # add intercept  
 y\_VS <- y[j]  
   
 # estimation of Beta given lambda [GAP]  
 BU\_hat <- c()  
   
 # predict the value  
 y\_hat <- X\_VS %\*% BU\_hat  
   
 # calculate the loss function (mean squared error) [GAP]  
 mse <-   
 MSE\_res[j, i] <- mse  
   
 }  
}  
  
# calculate the CV mean squared error  
CV\_mse <- colMeans(MSE\_res)

#### 4.

Plot the statistic given the values of lambda, and identify which value minimizes the function

#### 5.

Compare with the procedure implemented in the R package glmnet using this code

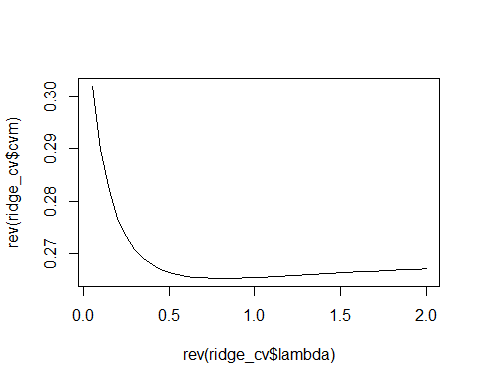
library(glmnet)

## Loaded glmnet 4.1-8

ridge\_cv <- cv.glmnet(x = cbind(M, X), y = y, lambda = lbd\_range,  
 type.measure = "mse", nfolds = 5, alpha = 0)

## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per  
## fold

plot(x = rev(ridge\_cv$lambda), y = rev(ridge\_cv$cvm), type = "l")



ridge\_cv$lambda.min

## [1] 0.8

#### Answer

#### 1.

# toy example  
y <- c(0.19, 1.23, 0.86, 1.23, 0.45)  
  
N <- length(y)  
  
X <- matrix(c(0,0,0,0,0,0,1,2,0,2,  
 1,0,0,1,1,1,2,1,0,1,  
 1,0,0,1,0,0,1,1,1,1,  
 1,1,1,1,0,1,2,1,1,1,  
 0,1,1,1,1,1,2,1,0,1), nrow = 5, byrow = TRUE)  
  
# identity matrix with dimension equal to the number of markers  
I <- diag(ncol(X))  
lambda <- 1  
M <- matrix(1, nrow = N, ncol = 1)  
  
MtM <- crossprod(M)  
MtX <- crossprod(M, X)  
XtM <- crossprod(X, M)  
XtX <- crossprod(X) + (I\*lambda)  
  
C <- rbind(cbind(MtM, MtX),  
 cbind(XtM, XtX))  
  
RHS <- rbind(crossprod(M, y),  
 crossprod(X, y))  
  
BU <- solve(C) %\*% RHS

#### 2.

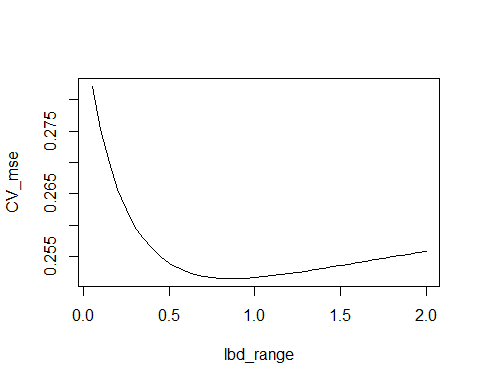
BU\_solve <- function(lbd = 1, M, Xm, y){  
   
 I <- diag(ncol(Xm))  
 MtM <- crossprod(M)  
 MtX <- crossprod(M, Xm)  
 XtM <- crossprod(Xm, M)  
 XtX <- crossprod(Xm) + (I\*lbd)  
   
 C <- rbind(cbind(MtM, MtX),  
 cbind(XtM, XtX))  
   
 RHS <- rbind(crossprod(M, y),  
 crossprod(Xm, y))  
   
 BU <- solve(C) %\*% RHS  
   
 return(BU)  
   
}  
  
lbd\_range <- seq(0.05, 2, 0.05)  
  
BU\_res <- lapply(X = lbd\_range, FUN = BU\_solve, M=M, Xm=X, y=y)  
  
BU\_res <- do.call(cbind, BU\_res)  
  
rownames(BU\_res) <- c("mu", paste0("B", 1:10))  
colnames(BU\_res) <- paste0("lbd = ", lbd\_range)

#### 3.

MSE\_res <- matrix(NA, nrow = 5, ncol = length(lbd\_range))  
  
for(i in 1:length(lbd\_range)){  
 for(j in 1:5){  
   
 # define TS and VS  
 M\_TS <- M[-j, , drop = FALSE]  
 X\_TS <- X[-j, , drop = FALSE]  
 y\_TS <- y[-j]  
   
 X\_VS <- cbind(1, X[j, , drop = FALSE]) # add intercept  
 y\_VS <- y[j]  
   
 # estimation of Beta given lambda  
 BU\_hat <- BU\_solve(M = M\_TS, X = X\_TS, y = y\_TS, lbd = lbd\_range[i])  
   
 # predict the value  
 y\_hat <- X\_VS %\*% BU\_hat  
   
 # calculate the loss function (mean squared error)  
 mse <- mean(sum((y\_VS - c(y\_hat))^2))  
 MSE\_res[j, i] <- mse  
   
 }  
}  
  
# calculate the CV mean squared error  
CV\_mse <- colMeans(MSE\_res)

#### 4.

plot(x = lbd\_range, y = CV\_mse, type = "l")



lbd\_min <- lbd\_range[which.min(CV\_mse)]  
lbd\_min

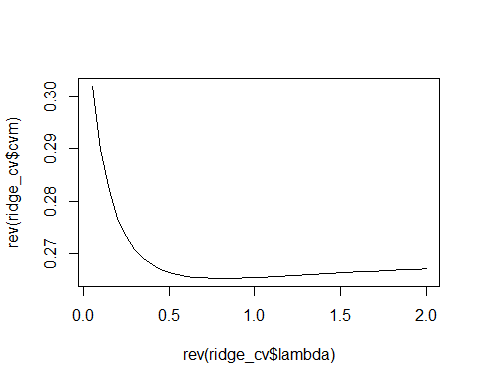
## [1] 0.85

#### 5.

library(glmnet)  
  
ridge\_cv <- cv.glmnet(x = cbind(M, X), y = y, lambda = lbd\_range,  
 type.measure = "mse", nfolds = 5, alpha = 0)

## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per  
## fold

plot(x = rev(ridge\_cv$lambda), y = rev(ridge\_cv$cvm), type = "l")



ridge\_cv$lambda.min

## [1] 0.8

#### Ex3: Regularization methods coefficients interpretation

#### 1.

Using the data below calculate the Ridge regression, LASSO, and Elastic net model with . Check that the properties associated to each model coefficients.

library(rrBLUP)  
  
load(file = "./data/ex\_geno\_BC08.RData")  
load(file = "./data/ex\_pheno\_BC08.RData")  
load(file = "./data/ex\_map\_BC08.RData")  
  
# calculate lambda  
X <- scale(geno)  
K <- X %\*% t(X) /ncol(X)  
d <- data.frame(GID = rownames(pheno), y = pheno[, 1])  
  
m <- kin.blup(data = d, geno = "GID", pheno = "y", K = K, GAUSS = FALSE)

#### 2.

Compare the distribution of the calculated parameters using the following code. What do you observe.

library(ggplot2)  
  
n\_methods <- 3  
n\_mk <- nrow(map)  
d\_res <- data.frame(chr = rep(map$chr, n\_methods),  
 pos.cM = rep(map$pos.cM, n\_methods),  
 method = rep(c("Ridge", "LASSO", "Enet"), each = n\_mk),  
 Beta = c(B\_Ridge$B[-1], B\_LASSO$B[-1], B\_Enet$B[-1]))  
  
d\_res$Beta <- abs(d\_res$Beta)  
d\_res$method <- factor(d\_res$method)  
  
p <- ggplot(d\_res, aes(x = pos.cM, y = Beta)) + geom\_line() +  
 facet\_grid(method ~ chr)

Do the same comparison adding estimated parameters from a GWAS analysis.

g\_data <- data.frame(map[, -3], t(geno), check.names = FALSE)  
y\_pheno <- data.frame(gid = rownames(pheno), y = pheno[, 1])  
  
B\_GWAS <- GWAS(pheno = y\_pheno, geno = g\_data, K = K)  
B\_GWAS <- B\_GWAS$y

#### Answer

#### 1.

library(rrBLUP)  
  
load(file = "./data/ex\_geno\_BC08.RData")  
load(file = "./data/ex\_pheno\_BC08.RData")  
load(file = "./data/ex\_map\_BC08.RData")  
  
# calculate lambda  
X <- scale(geno)  
K <- X %\*% t(X) /ncol(X)  
d <- data.frame(GID = rownames(pheno), y = pheno[, 1])  
  
m <- kin.blup(data = d, geno = "GID", pheno = "y", K = K, GAUSS = FALSE)  
Se <- m$Ve  
Sg <- m$Vg  
  
lbd <- Se/Sg  
  
# Ridge  
m\_Ridge <- glmnet(x = X, y = pheno[, 1], family = "gaussian",  
 alpha = 0, lambda = lbd)  
B\_Ridge <- data.frame(as.matrix(coef(m\_Ridge)))  
colnames(B\_Ridge)[1] <- "B"   
  
sum(B\_Ridge$B != 0)

## [1] 10001

# LASSO  
m\_LASSO <- glmnet(x = X, y = pheno[, 1], family = "gaussian",  
 alpha = 1, lambda = lbd)  
B\_LASSO <- data.frame(as.matrix(coef(m\_LASSO)))  
colnames(B\_LASSO)[1] <- "B"   
  
sum(B\_LASSO$B != 0)

## [1] 81

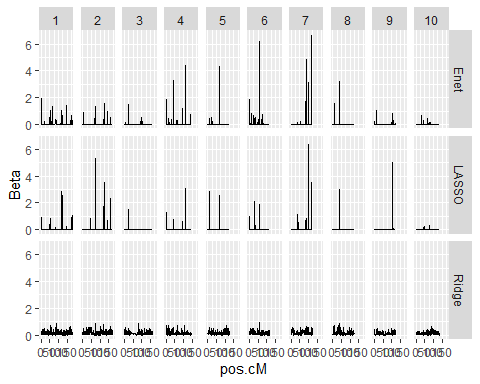
# Enet  
m\_Enet <- glmnet(x = X, y = pheno[, 1], family = "gaussian",  
 alpha = 0.5, lambda = lbd)  
B\_Enet <- data.frame(as.matrix(coef(m\_Enet)))  
colnames(B\_Enet)[1] <- "B"   
  
sum(B\_Enet$B != 0)

## [1] 385

The Ridge regression do not select any parameters (all parameters have a non-zero value). The LASSO select maximum non-zero parameters. The Elastic net select more than parameters.

#### 2.

# compare results between methods  
n\_methods <- 3  
n\_mk <- nrow(map)  
d\_res <- data.frame(chr = rep(map$chr, n\_methods),  
 pos.cM = rep(map$pos.cM, n\_methods),  
 method = rep(c("Ridge", "LASSO", "Enet"), each = n\_mk),  
 Beta = c(B\_Ridge$B[-1], B\_LASSO$B[-1], B\_Enet$B[-1]))  
  
d\_res$Beta <- abs(d\_res$Beta)  
d\_res$method <- factor(d\_res$method)  
  
p <- ggplot(d\_res, aes(x = pos.cM, y = Beta)) + geom\_line() +  
 facet\_grid(method ~ chr)  
p



The selected parameters are concentrated in specific regions (e.g. chromosome 7)

g\_data <- data.frame(map[, -3], t(geno), check.names = FALSE)  
y\_pheno <- data.frame(gid = rownames(pheno), y = pheno[, 1])  
  
# Classical GWAS with K (kinship correction)  
B\_GWAS <- rrBLUP::GWAS(pheno = y\_pheno, geno = g\_data, K = K)

## [1] "GWAS for trait: y"  
## [1] "Variance components estimated. Testing markers."

B\_GWAS <- B\_GWAS$y  
  
# comparison plot with other methods  
meth\_vec <- c("GWAS","Ridge", "LASSO", "Enet")  
n\_methods <- length(meth\_vec)  
d\_res <- data.frame(chr = rep(map$chr, n\_methods),  
 pos.cM = rep(map$pos.cM, n\_methods),  
 method = rep(meth\_vec, each = n\_mk),  
 Beta = c(B\_GWAS, B\_Ridge$B[-1], B\_LASSO$B[-1], B\_Enet$B[-1]))  
  
d\_res$Beta <- abs(d\_res$Beta)  
d\_res$method <- factor(d\_res$method, levels = meth\_vec)  
  
p <- ggplot(d\_res, aes(x = pos.cM, y = Beta)) + geom\_line() +  
 facet\_grid(method ~ chr)  
p

We can notice that the region of selected parameters correspond to the GWAS peak.

#### Ex4: Determination of using cross-validation

Using the cv.glmnet function determine the values that minimize the mean squared error for each method (Ridge, LASSO, Elastic net) using different size of k-fold with k = 5, 10, 20. Look at the distribution of the CV MSE function using plot. What can you observe.

#### Answer

# Ridge  
m\_cv\_5 <- cv.glmnet(x = X, y = pheno[, 1], family = "gaussian",  
 alpha = 0, nfolds = 5)  
m\_cv\_5$lambda.min

## [1] 12011.11

m\_cv\_10 <- cv.glmnet(x = X, y = pheno[, 1], family = "gaussian",  
 alpha = 0, nfolds = 10)  
m\_cv\_10$lambda.min

## [1] 11465.19

m\_cv\_20 <- cv.glmnet(x = X, y = pheno[, 1], family = "gaussian",  
 alpha = 0, nfolds = 20)  
m\_cv\_20$lambda.min

## [1] 10944.08

par(mfrow = c(2, 2))  
plot(m\_cv\_5, main = "Ridge k=5")  
plot(m\_cv\_10, main = "Ridge k=10")  
plot(m\_cv\_20, main = "Ridge k=20")  
  
  
# LASSO  
m\_cv\_5 <- cv.glmnet(x = X, y = pheno[, 1], family = "gaussian",  
 alpha = 1, nfolds = 5)  
m\_cv\_5$lambda.min

## [1] 5.706251

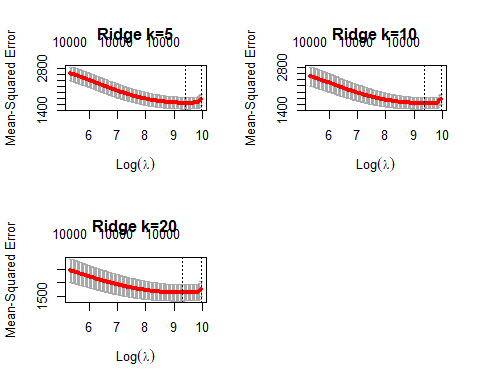
m\_cv\_10 <- cv.glmnet(x = X, y = pheno[, 1], family = "gaussian",  
 alpha = 1, nfolds = 10)  
m\_cv\_10$lambda.min

## [1] 5.199323

m\_cv\_20 <- cv.glmnet(x = X, y = pheno[, 1], family = "gaussian",  
 alpha = 1, nfolds = 20)  
m\_cv\_20$lambda.min

## [1] 6.560802

par(mfrow = c(2, 2))



plot(m\_cv\_5, main = "LASSO k=5")  
plot(m\_cv\_10, main = "LASSO k=10")  
plot(m\_cv\_20, main = "LASSO k=20")  
  
# Enet  
m\_cv\_5 <- cv.glmnet(x = X, y = pheno[, 1], family = "gaussian",  
 alpha = 0.5, nfolds = 5)  
m\_cv\_5$lambda.min

## [1] 15.80502

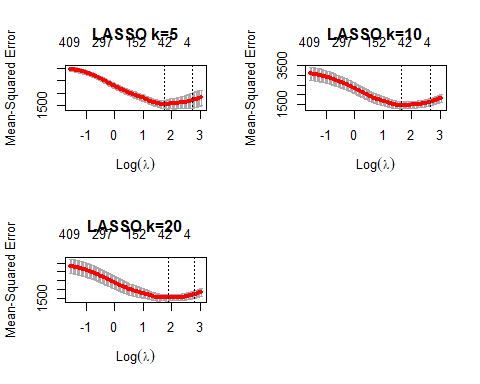
m\_cv\_10 <- cv.glmnet(x = X, y = pheno[, 1], family = "gaussian",  
 alpha = 0.5, nfolds = 10)  
m\_cv\_10$lambda.min

## [1] 9.044212

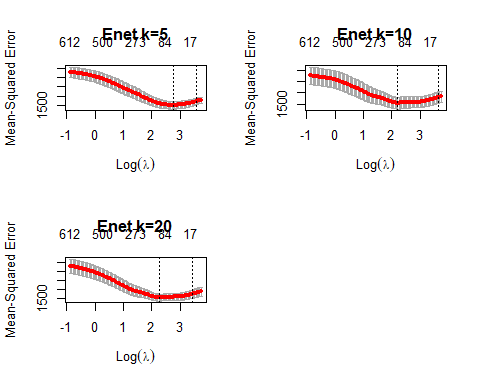
m\_cv\_20 <- cv.glmnet(x = X, y = pheno[, 1], family = "gaussian",  
 alpha = 0.5, nfolds = 20)  
m\_cv\_20$lambda.min

## [1] 9.474859

par(mfrow = c(2, 2))



plot(m\_cv\_5, main = "Enet k=5")  
plot(m\_cv\_10, main = "Enet k=10")  
plot(m\_cv\_20, main = "Enet k=20")

 The number of fold has an influence on the value of that minimize the prediction error however, it does not change a lot the number of selected parameters (top of the graph x axis).

#### Ex5: GBLUP with fixed effect from GWAS

Look at the code of the function that realize a GBLUP prediction after selecting peak from GWAS analysis to put them as fixed effect. Two methods are proposed: “QTL” and “RiceLipka”. What are the main differences between those methods

GBLUP\_GWAS\_fixed <- function(geno, GID, map, pheno, method = "QTL",  
 window = 10, nMksel = 10, threshold = 3,  
 verbose = FALSE, do\_plot = FALSE){  
   
 # format data  
 colnames(map) <- c("mk.names","chr", "pos.cM")  
   
 # a) GWAS ----  
 K <- A.mat(geno)  
 g\_data <- data.frame(map, t(geno), check.names = FALSE)  
 y\_pheno <- data.frame(gid = rownames(pheno), y = pheno[, 1])  
 B\_GWAS <- rrBLUP::GWAS(pheno = y\_pheno, geno = g\_data, K = K, plot = do\_plot)  
   
   
 # b) marker selection ----  
 if(method == "QTL"){  
 Q\_prof <- data.frame(mk.names = map[, 1], chr = map[, 2],  
 pos.ind = NA, pos.cM = map[, 3],  
 log10pval = B\_GWAS$y)  
 class(Q\_prof) <- c("QTLprof", "data.frame")  
 Qsel <- QTL\_select(Qprof = Q\_prof, threshold = threshold,  
 window = window, verbose = FALSE)  
 if(!is.null(Qsel)){  
 Q\_sel\_nm <- Qsel$mk.names  
 if(verbose){print(paste("Selected positions:", paste(Q\_sel\_nm, collapse = ", ")))}  
 } else {Q\_sel\_nm <- NULL}  
   
 } else if (method == "RiceLipka"){  
 B\_GWAS <- B\_GWAS[order(B\_GWAS$y, decreasing = TRUE), ]  
 Q\_sel\_nm <- B\_GWAS[, 1][1:nMksel]  
 if(verbose){print(paste("Selected positions:", paste(Q\_sel\_nm, collapse = ", ")))}  
 }  
   
 # c) geno matrix cleaning (window = 0 if method = "RiceLipka") ----  
 if(!is.null(Q\_sel\_nm)){  
   
 if(method == "RiceLipka"){  
 # win\_clean = 0  
 geno\_clean <- geno[, -which(map[, 1] %in% Q\_sel\_nm)]  
 } else if (method == "QTL"){  
   
 test.cof <- function(x, map, window) {  
   
 t1 <- map$chr == as.numeric(x[1])  
 t2 <- abs(map$pos.cM - as.numeric(x[2])) < window  
 !(t1 & t2)  
   
 }  
   
 mk.part <- apply(X = Qsel[, c(2, 4)], MARGIN = 1,  
 FUN = test.cof,  
 map = map, window = window)  
 sel\_mk <- apply(mk.part, 1, function(x) all(x))  
   
 geno\_clean <- geno[, sel\_mk]  
   
 }  
   
 } else{  
 geno\_clean <- geno  
 }  
   
 # d) GP ----  
   
 # calculate kinship  
 K <- A.mat(X = geno\_clean)  
 d <- data.frame(GID = GID, y = pheno)  
 colnames(d)[2] <- "y"  
   
 if(!is.null(Q\_sel\_nm)){  
 # add the fixed marker terms  
 d <- data.frame(d, geno[, which(map[, 1] %in% Q\_sel\_nm)])  
 colnames(d)[3:ncol(d)] <- Q\_sel\_nm  
   
 f\_formula <- paste0("y~1+", paste0(Q\_sel\_nm, collapse = "+"))  
 m <- mmer(fixed = as.formula(f\_formula),  
 random= ~vsr(GID,Gu=K),  
 rcov= ~ units,  
 data=d, verbose = FALSE, dateWarning = FALSE)  
   
 } else {  
   
 m <- tryCatch(mmer(fixed = y~1,  
 random= ~vsr(GID,Gu=K),  
 rcov= ~ units,  
 data=d, verbose = FALSE, dateWarning = FALSE),  
 error = function(x) NULL)  
   
   
 }  
   
 # return GP output  
 return(list(m = m, d = d, Q\_sel = Q\_sel\_nm))  
   
}

#### Answer

The “RiceLipka” method propose to select the n top markers. Those marker position are removed from the kinship before the calculation of the GBLUP with the selected markers as fixed effect.

The “QTL” method, select iteratively QTL positions that pass the threshold value with an exlusion window around those positions. The “QTL” method exlude the selected marker as well as the correlated positions in the neighborhood before calculating the kinship for the prediction.

#### Ex5: Genomic prediction illustration

We have now lear about two main class of prediction method: mixed model GBLUP and regularization regression (Ridge, LASSO, Elastic net) with possible variant like GBLUP with fixed effects. Look at the code below which represent a skeleton to realize a cross-validation evaluation. Complement this code with your favourite methods and software to compare the different method using the BC08 cross data.

library(mppR)  
library(rrBLUP)  
library(glmnet)  
library(ggplot2)  
  
load(file = "./data/ex\_geno\_BC08.RData")  
load(file = "./data/ex\_pheno\_BC08.RData")  
load(file = "./data/ex\_map\_BC08.RData")  
  
# elements for the models  
GID <- rownames(geno)  
map <- map[, -3]  
  
# calculate kinship  
K <- A.mat(X = geno)  
  
# process pheno data  
d <- data.frame(GID = rownames(pheno), y = pheno[, 2])  
pheno <- pheno[, 2, drop = FALSE]  
  
k <- 5  
n\_rep <- 2  
  
# space to store the results  
methods\_vec <- c("GLUP", "GBLUP-fixed", "Ridge", "LASSO", "Enet")  
n\_meth <- length(methods\_vec)  
res\_list <- vector(mode = "list", length = n\_meth)  
for(m in 1:n\_meth){  
 res\_list[[m]] <- cor\_res <- matrix(NA, nrow = k, ncol = n\_rep)  
}  
names(res\_list) <- methods\_vec   
  
for(j in 1:n\_rep){  
 # partition the data into k VS  
 VS <- matrix(sample(1:nrow(pheno)), ncol = k)  
 for(i in 1:k){  
   
 # make copy the original data that can be modified  
 d\_i <- d   
 d\_res <- d  
   
 # mask validation set data  
 d\_i$y[VS[, i]] <- NA  
   
 # calculate prediction model: GBLUP ----  
 m1 <- kin.blup(data = d\_i, geno = "GID", pheno = "y", K = K, GAUSS = FALSE)  
   
 # extract BLUP and add to result datset checking for order  
 u1 <- m1$g  
 d\_res$y\_pred1 <- u1[d$GID]  
   
 # calculate prediction model: GBLUP + fixed ----  
 m2 <- GBLUP\_GWAS\_fixed(geno = geno, GID = GID, map = map,  
 pheno = d\_i[, 2, drop = FALSE],  
 method = "RiceLipka", window = 10, nMksel = 10, threshold = 3,  
 verbose = FALSE, do\_plot = FALSE)  
   
 if(!is.null(m2$m)){  
 # fixed part  
 B\_hat <- m2$m$Beta$Estimate  
 names(B\_hat) <- m2$m$Beta$Effect  
 if(!is.null(m2$Q\_sel)){  
 X\_f <- cbind(rep(1, nrow(m2$d)), as.matrix(m2$d[, 3:ncol(m2$d)]))  
 colnames(X\_f)[1] <- "Int"  
 X\_f <- X\_f[, c('Int', names(B\_hat)[-1])]  
   
 } else {  
 X\_f <- matrix(rep(1, nrow(m2$d)), ncol = 1)  
 }  
 y\_hat\_fixed <- X\_f %\*% t(matrix(B\_hat, nrow = 1))  
   
 # random part   
 y\_hat\_geno <- m2$m$U$`u:GID`$y  
 d\_res$y\_pred2 <- c(y\_hat\_fixed + y\_hat\_geno[GID])  
   
 } else{d\_res$y\_pred2 <- NA}  
   
 # calculate prediction model: Ridge regression ----  
   
 # process the data  
 X\_TS <- geno[-VS[, i], ]  
 y\_TS <- d$y[-VS[, i]]  
   
 # remove NA values  
 NA\_pos <- is.na(y\_TS)  
 X\_TS <- X\_TS[!NA\_pos, ]  
 y\_TS <- y\_TS[!NA\_pos]  
   
 m3\_cv <- cv.glmnet()  
   
 # [...]  
   
 # calculate prediction model: LASSO ----  
   
 # [...]  
   
 # calculate prediction model: Enet ----  
   
 # [...]  
   
   
 colnames(d\_res) <- c("GID","y","GBLUP", "GB\_fix", "RR", "LASSO", "Enet")  
   
 # calculate the correlation (obs, pred)  
 d\_vs <- d\_res[VS[, i], ] # selection only VS data  
   
 # cor inter method  
 cor(d\_vs[, 2:7], use = "complete.obs")  
   
 # store the correlation results into list  
 for(m in 1:n\_meth){  
 res\_list[[m]][i, j] <- cor(x = d\_vs$y, y = d\_vs[, m + 2],  
 use = "complete.obs")  
 }  
   
 }  
}  
  
# Get the results ----  
  
# look at the results  
  
d <- c()  
for(i in 1:length(res\_list)){  
 d\_i <- data.frame(model = names(res\_list)[i], cor = c(res\_list[[i]]))  
 d <- rbind(d, d\_i)  
}  
  
pl <- ggplot(d, aes(x = model, y = cor)) +  
 geom\_boxplot() +   
 labs(title = "GP model comp", x = "models", y = "correlation(y\_obs, y\_pred)")  
  
pl

#### Answer

library(mppR)  
library(rrBLUP)  
library(glmnet)  
library(ggplot2)  
  
load(file = "./data/ex\_geno\_BC08.RData")  
load(file = "./data/ex\_pheno\_BC08.RData")  
load(file = "./data/ex\_map\_BC08.RData")  
  
# elements for the models  
GID <- rownames(geno)  
map <- map[, -3]  
  
# calculate kinship  
K <- A.mat(X = geno)  
  
# process pheno data  
d <- data.frame(GID = rownames(pheno), y = pheno[, 2])  
pheno <- pheno[, 2, drop = FALSE]  
  
k <- 5  
n\_rep <- 2  
  
# space to store the results  
methods\_vec <- c("GLUP", "GBLUP-fixed", "Ridge", "LASSO", "Enet")  
n\_meth <- length(methods\_vec)  
res\_list <- vector(mode = "list", length = n\_meth)  
for(m in 1:n\_meth){  
 res\_list[[m]] <- cor\_res <- matrix(NA, nrow = k, ncol = n\_rep)  
}  
names(res\_list) <- methods\_vec   
  
for(j in 1:n\_rep){  
 VS <- matrix(sample(1:nrow(pheno)), ncol = k)  
 for(i in 1:k){  
   
 # prepare data  
 d\_i <- d # copy the original data  
 d\_res <- d  
   
 # mask validation set data  
 d\_i$y[VS[, i]] <- NA  
   
 # calculate prediction model: GBLUP ----  
 m1 <- kin.blup(data = d\_i, geno = "GID", pheno = "y", K = K, GAUSS = FALSE)  
   
 # extract BLUP and add to datset checking for order  
 u1 <- m1$g  
 d\_res$y\_pred1 <- u1[d$GID]  
   
 # calculate prediction model: GBLUP + fixed ----  
 m2 <- GBLUP\_GWAS\_fixed(geno = geno, GID = GID, map = map,  
 pheno = d\_i[, 2, drop = FALSE],  
 method = "RiceLipka", window = 10, nMksel = 10, threshold = 3,  
 verbose = FALSE, do\_plot = FALSE)  
   
 if(!is.null(m2$m)){  
 # fixed part  
 B\_hat <- m2$m$Beta$Estimate  
 names(B\_hat) <- m2$m$Beta$Effect  
 if(!is.null(m2$Q\_sel)){  
 X\_f <- cbind(rep(1, nrow(m2$d)), as.matrix(m2$d[, 3:ncol(m2$d)]))  
 colnames(X\_f)[1] <- "Int"  
 X\_f <- X\_f[, c('Int', names(B\_hat)[-1])]  
   
 } else {  
 X\_f <- matrix(rep(1, nrow(m2$d)), ncol = 1)  
 }  
 y\_hat\_fixed <- X\_f %\*% t(matrix(B\_hat, nrow = 1))  
   
 # random part   
 y\_hat\_geno <- m2$m$U$`u:GID`$y  
 d\_res$y\_pred2 <- c(y\_hat\_fixed + y\_hat\_geno[GID])  
   
 } else{d\_res$y\_pred2 <- NA}  
   
 # calculate prediction model: Ridge regression ----  
 X\_TS <- geno[-VS[, i], ]  
 y\_TS <- d$y[-VS[, i]]  
   
 # remove NA values  
 NA\_pos <- is.na(y\_TS)  
 X\_TS <- X\_TS[!NA\_pos, ]  
 y\_TS <- y\_TS[!NA\_pos]  
   
 m3\_cv <- cv.glmnet(x = X\_TS, y = y\_TS, family = "gaussian",  
 alpha = 0, nfolds = 10)  
   
 m3\_fit <- glmnet(x = X\_TS, y = y\_TS, family = "gaussian",  
 alpha = 0, lambda = m3\_cv$lambda.min)  
 B\_m3 <- m3\_fit$beta  
   
 # calculate BLUP (X \* B) add to datset checking for order  
 u3 <- as.matrix(geno %\*% B\_m3)  
 d\_res$y\_pred3 <- u3[d$GID, 1]  
   
 # calculate prediction model: LASSO ----  
   
 m4\_cv <- cv.glmnet(x = X\_TS, y = y\_TS, family = "gaussian",  
 alpha = 0.5, nfolds = 10)  
   
 m4\_fit <- glmnet(x = X\_TS, y = y\_TS, family = "gaussian",  
 alpha = 0.5, lambda = m4\_cv$lambda.min)  
 B\_m4 <- m4\_fit$beta  
   
 # calculate BLUP (X \* B) add to datset checking for order  
 u4 <- as.matrix(geno %\*% B\_m4)  
 d\_res$y\_pred4 <- u4[d$GID, 1]  
   
 # calculate prediction model: Enet ----  
 m5\_cv <- cv.glmnet(x = X\_TS, y = y\_TS, family = "gaussian",  
 alpha = 1, nfolds = 10)  
   
 m5\_fit <- glmnet(x = X\_TS, y = y\_TS, family = "gaussian",  
 alpha = 1, lambda = m5\_cv$lambda.min)  
 B\_m5 <- m5\_fit$beta  
   
 # calculate BLUP (X \* B) add to datset checking for order  
 u5 <- as.matrix(geno %\*% B\_m5)  
 d\_res$y\_pred5 <- u3[d$GID, 1]  
   
 colnames(d\_res) <- c("GID","y","GBLUP", "GB\_fix", "RR", "LASSO", "Enet")  
   
 # calculate the correlation (obs, pred)  
 d\_vs <- d\_res[VS[, i], ] # selection only VS data  
   
 # cor inter method  
 cor(d\_vs[, 2:7], use = "complete.obs")  
   
   
 # print(plot(x = d\_vs$y, y = d\_vs$y\_pred, xlab = "obs", ylab = "pred",  
 # main = paste("Rep", j, paste0("k=",i))))  
   
 for(m in 1:n\_meth){  
 res\_list[[m]][i, j] <- cor(x = d\_vs$y, y = d\_vs[, m + 2],  
 use = "complete.obs")  
 }  
   
 }  
}  
  
# Get the results ----  
  
# look at the results  
  
d <- c()  
for(i in 1:length(res\_list)){  
 d\_i <- data.frame(model = names(res\_list)[i], cor = c(res\_list[[i]]))  
 d <- rbind(d, d\_i)  
}  
  
pl <- ggplot(d, aes(x = model, y = cor)) +  
 geom\_boxplot() +   
 labs(title = "GP model comp", x = "models", y = "correlation(y\_obs, y\_pred)")  
  
pl