WorkShop Genomic Evaluation

Marie PEGARD

05/11/2020

Contents

** Please Note that scripts are not optimized: effort was made to make explicit steps for clarity, following a parsimonious way rather than fast implicit style **

The first chunk is used to define the document settings. The options are as follows:

- The figures are centered
- The image quality is defined for high definition screens.
- The width corresponds to about 7 inch the width of the A4 page.
- Only the last figure is kept when composing complex figures (via loops for example).
- The figures are stored under the "png" format and in the "Figures" folder with the prefix "Figure_" before the chunk title.
- The two tidy option are to prevent the script text from going beyond the margins of the page
- the option *warning* = *F* allow to remove the warnings from the output
- The last two options are to save the chunk output and not have to redo everything for a small change. Be careful if the change has a big impact for the next chuncks you have to delete the cache of the next chunks and restart (The cache files are located in the following folder : "WorkShop_cache").

Package installation and Data Loading

Package Installation and loading

First, we will load and install the packages needed for the analyses. Some packages are not available on CRAN and need to be installed in another way. The package *LDheatmap* is needed to use *Gapit3* but it is not available for the R version on Progeno. It is necessary to install manually an older version.

When you install, choose the option to install all the dependencies. If the packages below are already installed they will not be reinstalled. because the script tests the presence of the package before installation.

```
if (!"anyLib" %in% installed.packages()) {
    install.packages("anyLib")
}

if (!"breedR" %in% installed.packages()) {
    devtools::install_github("famuvie/breedR")
}

if (!"emma" %in% installed.packages()) {
    html = "https://github.com/Gregor-Mendel-Institute/mlmm/files/1356516/emma_1.1.2.tar.gz"
    install.packages(html, repos = NULL)
}

if (!requireNamespace("BiocManager", quietly = TRUE)) {
    install.packages("BiocManager")
}

if (!"snpStats" %in% installed.packages()) {
    BiocManager::install("snpStats")
}

if (!"LDheatmap" %in% installed.packages()) {
    html = "https://cran.r-project.org/src/contrib/Archive/LDheatmap/LDheatmap_0.99-7.tar.gz"
    install.packages(html, repos = NULL)
}

if (!"mlmm" %in% installed.packages()) {
    devtools::install_github("Gregor-Mendel-Institute/MultLocMixMod")
}
```

The package list below are packages present on the CRAN or installed in the previous step. Their loading is necessary for the continuation of the script. If one of the packages is missing, it will be installed via the anyLib command.

The anyLib function, checks if the packages are installed and updated, if not it tries to install it (or a new version) and if yes it loads the package. Returns a vector of TRUE or FALSE with the name of the package.

```
lib <- c("ggplot2", "viridis", "breedR", "apercu", "dplyr", "tidyr", "GAPIT3",
    "data.table", "plyr", "gplots", "qtl", "nadiv", "rrBLUP", "BGLR", "corpcor",
    "ggpubr", "emma", "mlmm", "FactoMineR", "factoextra")

anyLib::anyLib(lib)
```

```
##    ggplot2    viridis     breedR     apercu      dplyr      tidyr     GAPIT3
##       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE
## data.table       plyr     gplots        qtl      nadiv     rrBLUP       BGLR
##       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE
##    corpcor     ggpubr       emma       mlmm FactoMineR factoextra
##       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE
```

I created some useful fonctions that I wrote in file name *functions.R*, use the fonction "source" to upload them in the global environment.

```
source("functions.R")
```

Introduction

Genome-wide association studies (GWAS), genomic evaluation (GWE) and genomic selection (GS) have spread rapidly throughout the world in both plant and animal species. This workshop covers the preliminary stages of data preparation, QTL detection, genomic evaluation and finally genomic selection. Presenting all the existing methods and techniques to you would be too long and would require a week rather than a day. I have therefore selected a few methods that are easy to implement and that have proven their robustness.

Obervation and understanding of data

A good understanding of an experimental design and the collected data observation takes part of the experiment. This is an essential pre-requisite for the correct and relevant preparation of field data for further analysis. In this part, I provide guidelines on how to observe and analyse field data, on the preparation steps, and on how to fit a linear model to fit the data for analysis in GWAS, GWE or GS.

The aim of genome-wide association studies, for example, is to determine the relationships between phenotypes and genotypes. In this case, rare phenotypes and their associated genotypes are of great interest. These are more often the result of technical errors or bad models rather than genetic reality. This is why it is important to bring the same rigour to the analysis of phenotypic and genotypic data. To do this, linear models are a good way of extracting the components of the phenotype. An overview of the process is provided in the figure bellow.

Loading Dataset

From Progeno (Progeno tutorial)

If you used the Progeno graphical interface the dataset is already in the global environment

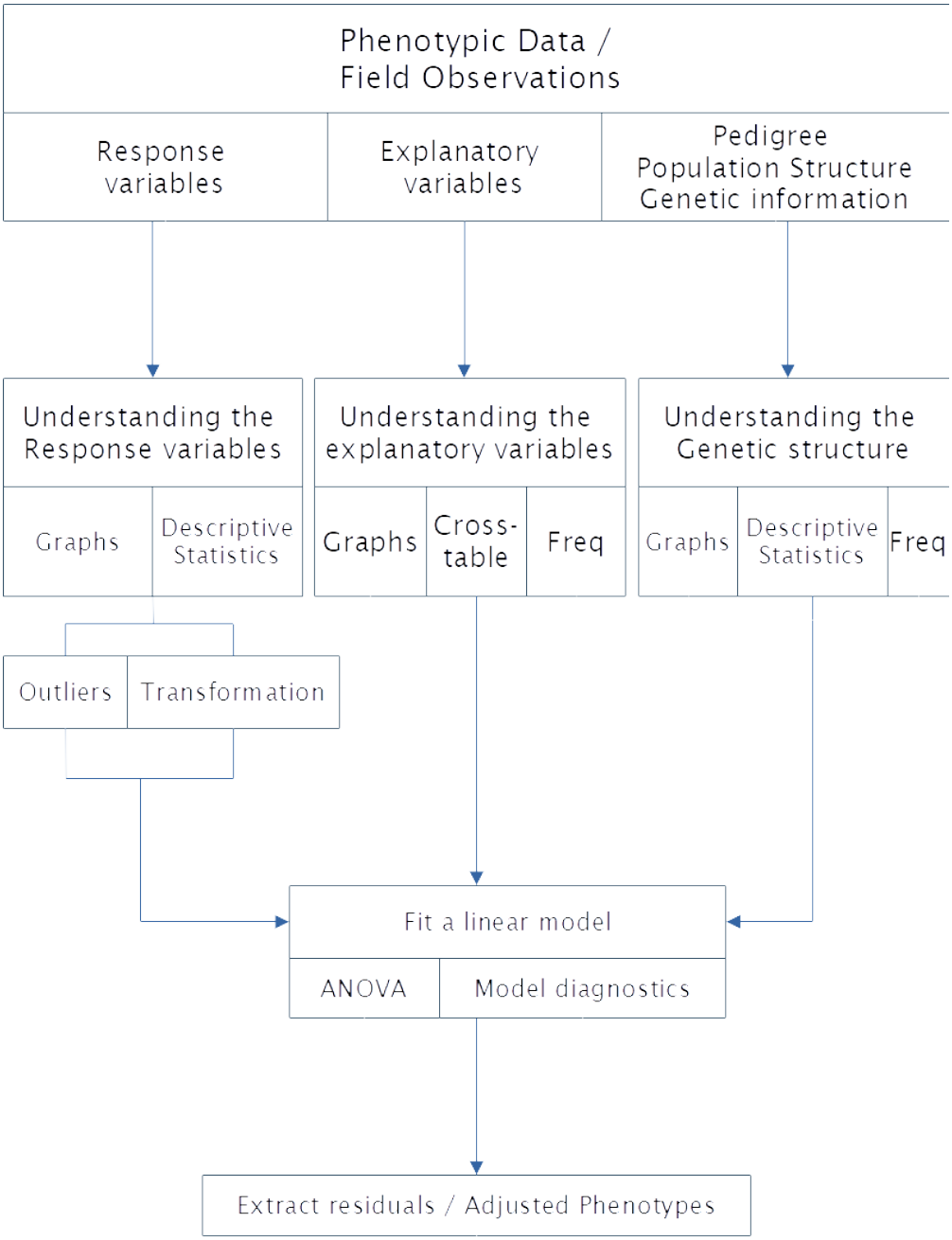


Figure 1: Phenotypic data process adapted from Gondro et al., 2019

From a dataset already saved in the Progeno's server

```
load(file = "DataSet/genotypes_soja.RData")
load(file = "DataSet/Phenotypes.RData")
```

From a file in your computer (slide number 103)

Once you have followed the steps on the tutorial from page X to Y you will be able to load the data in your working environment.

```
geno <- fread("DataSet/euclegnewblastsorted.vcf", stringsAsFactors = F)  #read the file and load it
ped <- fread("DataSet/Pedigree_Soja.txt", header = T,
    na.strings = "")

pheno18 <- fread("DataSet/IFVCNS 2018.txt", header = T,
    na.strings = "NA", dec = ",", stringsAsFactors = F)
pheno18$Trial <- "IFVCNS"  # Create a column and fill it with the character string 'IFVCNS'.
pheno18$Year <- 2018  # Create a column and fill it with 2018.

pheno19 <- fread("DataSet/IFVCNS 2019.txt", header = T,
    na.strings = "NA", dec = ",", stringsAsFactors = F)
pheno19$Trial <- "IFVCNS"
pheno19$Year <- 2019

genoet_struc <- fread("DataSet/Genetic_structure.txt",
    header = T, na.strings = "NA", stringsAsFactors = F)
```

Dataset Visualization and Preparation

The graphical display of data is a powerful tool for : - detecting errors - detect the need to transform them - also provides a visual tool for exploring the relationships between variables.

Typically in the data preparation stage, histograms, xy point clouds and box plots are used for visualization.

Phenotypes

For phenotyping I combine the two datasets to have only one object.

```
phenoSoy <- rbind(pheno18, pheno19)  #Concatenates data by lines
rm(pheno18, pheno19)
```

Descriptive statistics summarize data to facilitate the exploration of data characteristics. Descriptive statistics include measures of centrality and dispersion. Measures of centrality include the mean, mode, and median. Measures of dispersion include the standard deviation and the range.

```
summary(phenoSoy)  # Basic statistics
```

```
##       Plot            Row            Column     EntryNo
##  Min.   :  1.0   Min.   : 1.00   Min.   :1   Min.   :  1.00
##  1st Qu.:108.8   1st Qu.:12.75   1st Qu.:3   1st Qu.: 90.75
##  Median :216.5   Median :24.50   Median :5   Median :180.50
##  Mean   :216.5   Mean   :24.50   Mean   :5   Mean   :174.37
```

```
##  3rd Qu.:324.2   3rd Qu.:36.25   3rd Qu.:7   3rd Qu.:270.25
##  Max.   :432.0   Max.   :48.00   Max.   :9   Max.   :360.00
##
##    EUCLEGID         Accessionname      Plantemergence        R1
##  Length:864         Length:864         Min.   :1.000   Min.   :24.00
##  Class :character   Class :character   1st Qu.:7.000   1st Qu.:40.00
##  Mode  :character   Mode  :character   Median :9.000   Median :60.00
##                                        Mean   :8.046   Mean   :54.09
##                                        3rd Qu.:9.000   3rd Qu.:65.00
##                                        Max.   :9.000   Max.   :88.00
##                                                        NA's   :3
##        R2              R5              R8          Seedyield    Proteincontent
##  Min.   :33.00   Min.   :35.00   Min.   : 78.0   Min.   : 160   Min.   :36.90
##  1st Qu.:43.00   1st Qu.:47.00   1st Qu.:111.0   1st Qu.:1773   1st Qu.:41.09
##  Median :62.00   Median :64.00   Median :123.0   Median :2300   Median :42.13
##  Mean   :56.08   Mean   :60.93   Mean   :122.2   Mean   :2329   Mean   :42.34
##  3rd Qu.:67.00   3rd Qu.:72.00   3rd Qu.:134.0   3rd Qu.:2853   3rd Qu.:43.41
##  Max.   :88.00   Max.   :93.00   Max.   :166.0   Max.   :4827   Max.   :49.48
##  NA's   :3       NA's   :3                                      NA's   :41
##     Trial                Year
##  Length:864         Min.   :2018
##  Class :character   1st Qu.:2018
##  Mode  :character   Median :2018
##                     Mean   :2018
##                     3rd Qu.:2019
##                     Max.   :2019
##
```

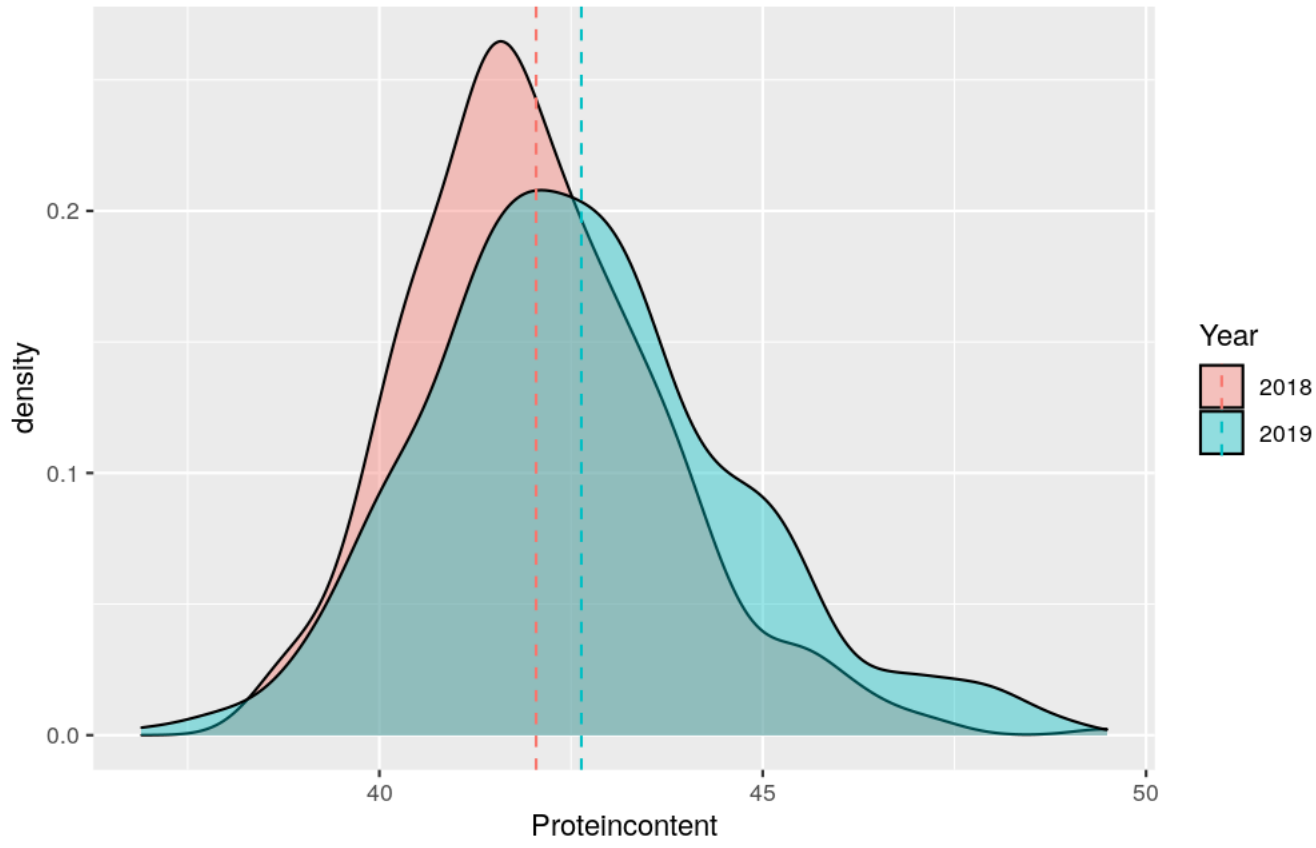
I look at the distribution and the phenotypic mean of the traits depending on the year.

```
summary(phenoSoy)  # Basic statistics
```

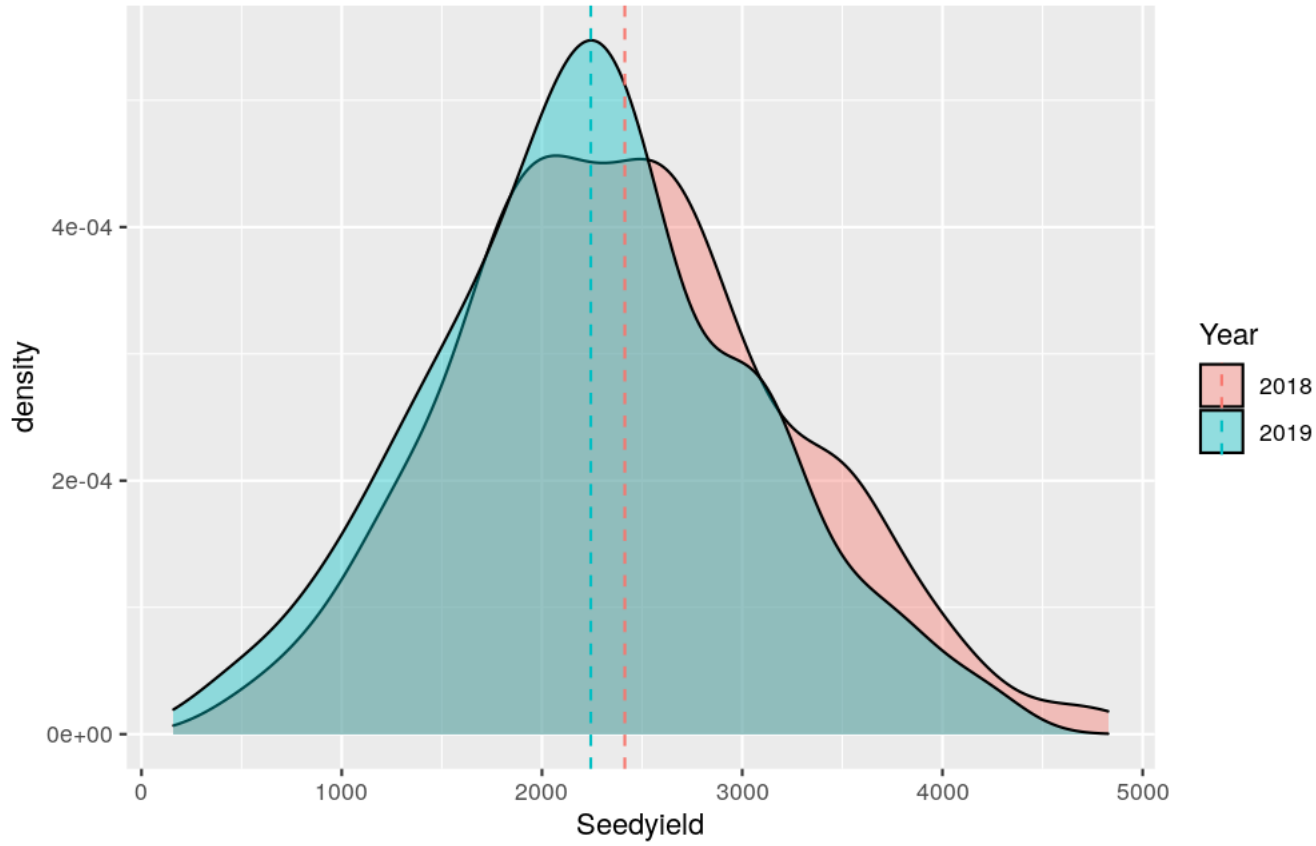
```
##       Plot            Row            Column     EntryNo
##  Min.   :  1.0   Min.   : 1.00   Min.   :1   Min.   :  1.00
##  1st Qu.:108.8   1st Qu.:12.75   1st Qu.:3   1st Qu.: 90.75
##  Median :216.5   Median :24.50   Median :5   Median :180.50
##  Mean   :216.5   Mean   :24.50   Mean   :5   Mean   :174.37
##  3rd Qu.:324.2   3rd Qu.:36.25   3rd Qu.:7   3rd Qu.:270.25
##  Max.   :432.0   Max.   :48.00   Max.   :9   Max.   :360.00
##
##    EUCLEGID         Accessionname      Plantemergence        R1
##  Length:864         Length:864         Min.   :1.000   Min.   :24.00
##  Class :character   Class :character   1st Qu.:7.000   1st Qu.:40.00
##  Mode  :character   Mode  :character   Median :9.000   Median :60.00
##                                        Mean   :8.046   Mean   :54.09
##                                        3rd Qu.:9.000   3rd Qu.:65.00
##                                        Max.   :9.000   Max.   :88.00
##                                                        NA's   :3
##        R2              R5              R8          Seedyield    Proteincontent
##  Min.   :33.00   Min.   :35.00   Min.   : 78.0   Min.   : 160   Min.   :36.90
##  1st Qu.:43.00   1st Qu.:47.00   1st Qu.:111.0   1st Qu.:1773   1st Qu.:41.09
##  Median :62.00   Median :64.00   Median :123.0   Median :2300   Median :42.13
##  Mean   :56.08   Mean   :60.93   Mean   :122.2   Mean   :2329   Mean   :42.34
##  3rd Qu.:67.00   3rd Qu.:72.00   3rd Qu.:134.0   3rd Qu.:2853   3rd Qu.:43.41
##  Max.   :88.00   Max.   :93.00   Max.   :166.0   Max.   :4827   Max.   :49.48
```

```
##  NA's   :3       NA's   :3                                      NA's   :41
##     Trial                Year
##  Length:864         Min.   :2018
##  Class :character   1st Qu.:2018
##  Mode  :character   Median :2018
##                     Mean   :2018
##                     3rd Qu.:2019
##                     Max.   :2019
##
```

```
# Calculates an average per year
mu <- ddply(phenoSoy, "Year", summarise, grp.mean = mean(Proteincontent,
    na.rm = T))

ggplot(phenoSoy, aes(x = Proteincontent, fill = as.character(Year))) +
    geom_density(alpha = 0.4) + geom_vline(data = mu,
    aes(xintercept = grp.mean, color = as.character(Year)),
    linetype = "dashed") + labs(fill = "Year", color = "Year")
```




```
mu <- ddply(phenoSoy, "Year", summarise, grp.mean = mean(Seedyield,
    na.rm = T))

ggplot(phenoSoy, aes(x = Seedyield, fill = as.character(Year))) +
    geom_density(alpha = 0.4) + geom_vline(data = mu,
    aes(xintercept = grp.mean, color = as.character(Year)),
    linetype = "dashed") + labs(fill = "Year", color = "Year")
```




```
rm(mu)
```

Observation of the phenotypic data shows that the annual averages are very close but not identical. In addition, the distribution of phenotypes differs. This indicates a year and or environmental effect.

Genotypes

Looking at the size and format of the objects is important to check that there are no errors in the previous manipulation or to make the appropriate modifications. Here we can see that we have 226798 markers and 486 individuals.

```
dim(geno)  # gives the number of row and column
```

```
## [1] 226798    486
```

```
str(geno[, 1:15])  # Have a look to the structure of the dataset
```

```
## Classes 'data.table' and 'data.frame':   226798 obs. of  15 variables:
##  $ CHROM         : chr  "Gm01" "Gm01" "Gm01" "Gm01" ...
##  $ POS           : int  11839 24952 26003 29671 30712 30765 32323 36434 37018 38482 ...
##  $ ID            : chr  "AX-93664820" "AX-93912564" "AX-93616364" "AX-93616449" ...
##  $ REF           : chr  "C" "A" "T" "A" ...
##  $ ALT           : chr  "G" "G" "C" "G" ...
##  $ QUAL          : chr  "." "." "." "." ...
##  $ FILTER        : chr  "." "." "." "." ...
##  $ INFO          : chr  "." "." "." "." ...
```

```
##  $ FORMAT        : chr  "GT" "GT" "GT" "GT" ...
##  $ EUC_GM_001.CEL: chr  "1/1" "0/0" "1/1" "0/0" ...
##  $ EUC_GM_002.CEL: chr  "1/1" "0/0" "1/1" "0/0" ...
##  $ EUC_GM_003.CEL: chr  "1/1" "0/0" "1/1" "0/0" ...
##  $ EUC_GM_004.CEL: chr  "1/1" "0/0" "1/1" "0/0" ...
##  $ EUC_GM_005.CEL: chr  "1/1" "1/1" "0/0" "1/1" ...
##  $ EUC_GM_006.CEL: chr  "1/1" "0/0" "1/1" "0/0" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

We observe that the suffix ".CEL" is attached to the names of individuals, which will interfere to connect the phenotypes with the genotyping. The suffix is removed from the column's names.

```
colnames(geno) <- sub(pattern = "[.]CEL", replacement = "",
    x = colnames(geno))
# Remove the pattern '.CEL' from the colnames. To
# remove the '.' use [.] because '.' in regular
# expression means any character.
colnames(geno)[1:20]
```

```
##  [1] "CHROM"      "POS"        "ID"         "REF"        "ALT"
##  [6] "QUAL"       "FILTER"     "INFO"       "FORMAT"     "EUC_GM_001"
## [11] "EUC_GM_002" "EUC_GM_003" "EUC_GM_004" "EUC_GM_005" "EUC_GM_006"
## [16] "EUC_GM_007" "EUC_GM_008" "EUC_GM_009" "EUC_GM_010" "EUC_GM_011"
```

```
# Check if the pattern have been removed. Yes !
```

In this case, some columns are useless for the following steps, so they are deleted. The columns containing the position of the markers are kept in another object. The geno object will only contain genotyping data.

```
# The unnecessary columns are removed
geno$REF <- NULL
geno$ALT <- NULL
geno$QUAL <- NULL
geno$FILTER <- NULL
geno$INFO <- NULL
geno$FORMAT <- NULL

# Split the data into Map and Genotypes
Map <- geno[, c("ID", "CHROM", "POS")]
rownames(geno) <- geno$ID
geno$ID <- NULL
geno$POS <- NULL
geno$CHROM <- NULL
```

For our analyses, the current genotyping format is not suitable. We need a numerical format. This can be 0,1,2 or 0,0.5,1 or -1,0,-1 to represent the allelic doses. A first step is to recode it and to ensure that the recoding has worked properly. For this I use a function of my creation (*replace_anychar*) that allows to do it quickly. You have a description and an example in the file "functions.R".

```
# Recoding the genotyping data in -1,0,1 it could
# also be 0,1,2 or 0, 0.5 ,1

# Verification step we look at the numbers in each
# category of genotyoage

# Before recoding
```

```
# sum(geno == '1/1') -> 55037892

# sum(geno == '0/1') -> 745670

# sum(geno == '1/0') -> 0

# sum(geno == '0/0') -> 52170833

# sum(geno == './.') -> 228251

geno_recode <- geno

geno_recode <- replace_anychar(geno_recode, "1/1",
    "1")
geno_recode <- replace_anychar(geno_recode, "0/1",
    "0")
geno_recode <- replace_anychar(geno_recode, "0/0",
    "-1")
geno_recode <- replace_anychar(geno_recode, "./.",
    NA)

geno_recode <- apply(geno_recode, 2, as.numeric)

rownames(geno_recode) <- rownames(geno)
ap(geno_recode)
```

```
##             EUC_GM_001 EUC_GM_002 EUC_GM_003 EUC_GM_004 EUC_GM_005
## AX-93664820          1          1          1          1          1
## AX-93912564         -1         -1         -1         -1          1
## AX-93616364          1          1          1          1         -1
## AX-93616449         -1         -1         -1         -1          1
## AX-93616475          1          1          1          1          1
```

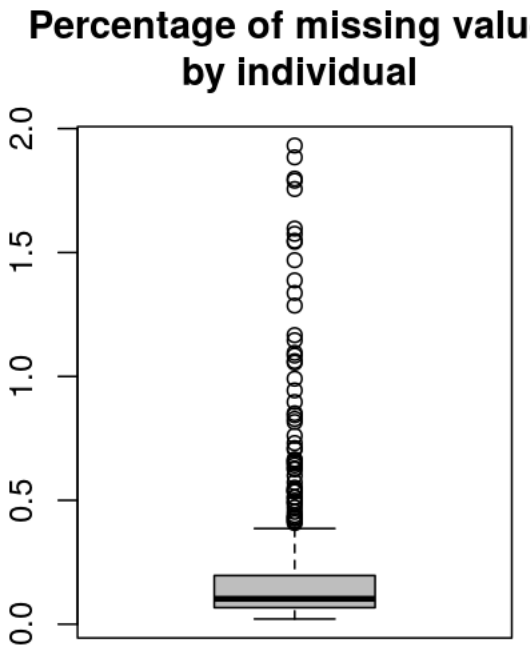
```
# After recoding sum(geno_recode == 1,na.rm = T) ->
# 55037892 sum(geno_recode == 0,na.rm = T) ->
# 745670 sum(geno_recode == -1,na.rm = T) ->
# 55037892 sum(is.na(geno_recode) ) -> 228251

rm(geno)
```

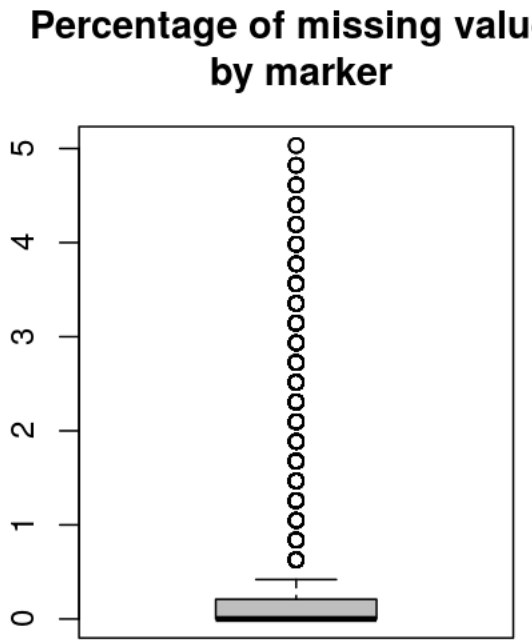
An important aspect in genotyping data is missing data. Indeed, when there is too much missing data, it is necessary to remove markers or individuals from the analysis. If there are a reasonable number of missing data, several solutions are possible. First of all, imputation software (BEAGLE, FImpute, AlphaImpute) can be used, or imputing with the mean or allelic frequency. There is no fixed threshold for determining whether the number of missing data is reasonable or not. It all depends on the population of interest; when one has access to pedigree and full genotyping in the parents, it is possible to impute almost 80% or even 90% of the data in the offspring, for completely independent individuals, beyond 1% or 2% of the population, the bias brought about by imputation can be critical.

Visualisation of missing data by markers and by individuals/population

```
#sum(is.na(geno_recode))/(dim(geno_recode)[[1]]*dim(geno_recode)[[2]])*100
par(mfrow=c(1,2))
boxplot(colMeans(is.na(geno_recode))*100,col="grey",
```

```
        main = "Percentage of missing value \n by individual")
boxplot(rowMeans(is.na(geno_recode))*100,col="grey",

        main = "Percentage of missing value \n by marker")
```



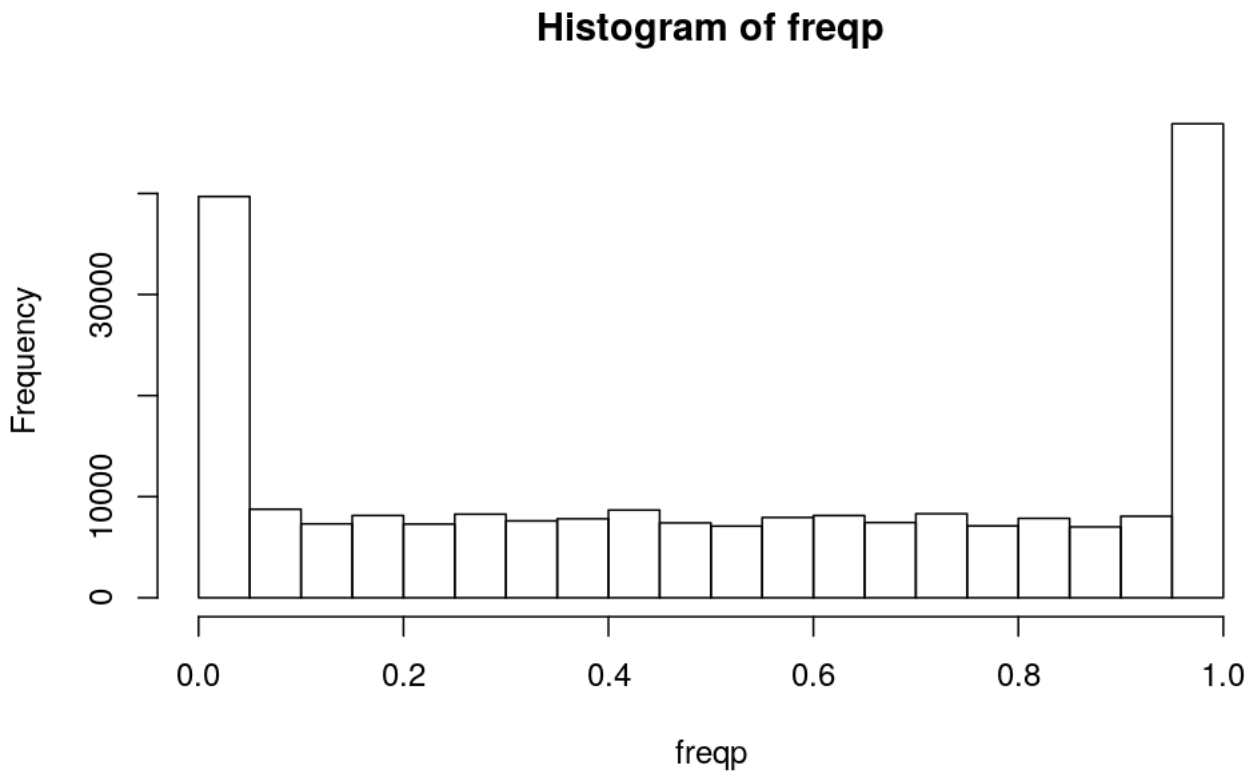



```
par(mfrow=c(1,1))
```

In the present situation, the number of missing value is really low (0.2109867%) .

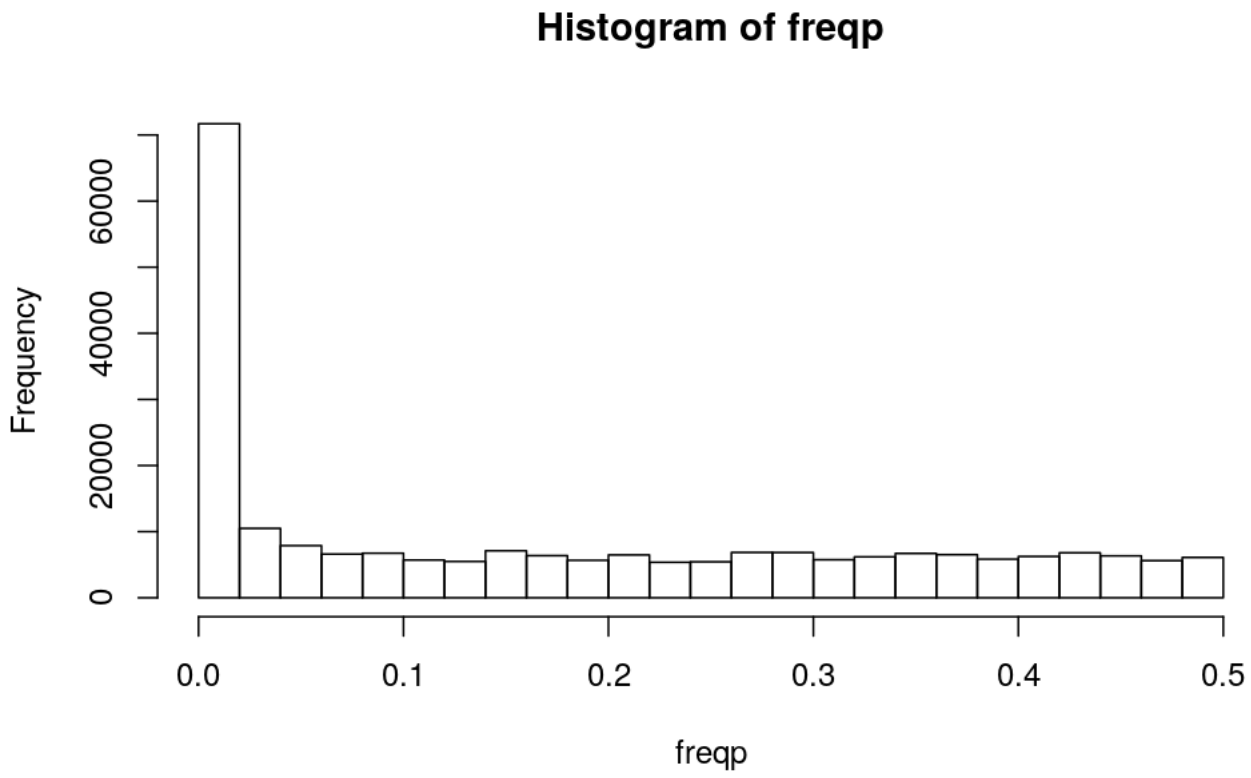
I choosed to replace them by the allelic frequency of the overall population, marker by marker.

```
M <- t(geno_recode[, colnames(geno_recode) %in% phenoSoy$EUCLEGID])
# I transpose the matrice to have the individuals
# in row and the markers in column.

freqp <- (colMeans(M, na.rm = T)/2) + 0.5
# Estimation of the allelic frequency when the
# genotyping is -1, 0, 1
hist(freqp)
```



Conversion of the allelic frequency to the minor allelic frequency (MAF)

```
freqp[freqp > 0.5] <- 1 - freqp[freqp > 0.5]
# Conversion of the allelic frequency to the minor
# allelic frequency

hist(freqp)
```

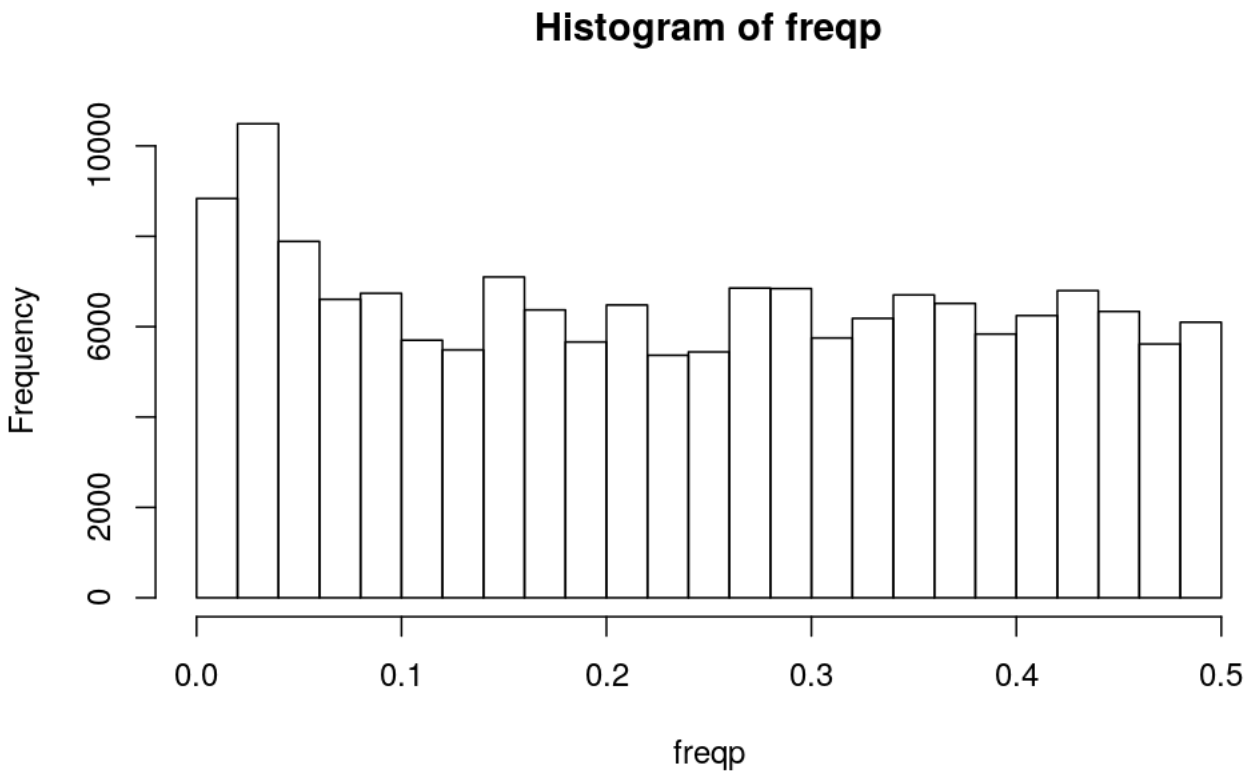


We remove SNPs with a MAF of less than 0.01 (62,863 SNPs are concerned) because at this frequency a very large number of individuals are needed to be able to estimate their effect correctly. To let them bias the analysis would be a mistake.

```
table(freqp <= 0.01)  # 62863 markers have a MAF lower than 0.01
```

```
##
##  FALSE   TRUE
## 163935  62863
```

```
Mapinf1perc <- names(which(freqp <= 0.01))
# I keep the deleted marker names

freqp <- freqp[freqp > 0.01]
hist(freqp)
```



We showed above that there were some missing data. These amount to `0.2306412`%, which is low. We can afford to replace the missing data by mean allelic frequency because this will not lead to a significant bias in the next steps. In the case of a significantly higher number of missing data, consideration should be given to removing the markers and/or turning to imputation methods.

The method I use to replace missing data with the allelic frequency is as follows:

- I remove markers with an allelic frequency lower than 1%.
- I create a matrix P of the same size as M and fill it with the allelic frequency.
- I duplicate the matrices M and P and obtain the matrices Mbis and Pbis
- I replace the positions where M has no missing data with zero in the Pbis matrix
- I replace the missing data with zero in the Mbis matrix
- I add the two marices Mbis and Pbis to keep the original genotyping and put the allelic frequency in place of the missing data.

```
dim(M)
```

```
## [1]    357 226798
```

```
M <- M[, !colnames(M) %in% Mapinf1perc]
# I remove the marker with a MAF lower than 0.01
# from the matrix
dim(M)
```

```
## [1]    357 163935
```

```
# I create two matrix of the same size as M filled
# with the allelic frequencies by columns.
ap(freqp)
```

```
## AX-93664820 AX-93912564 AX-93616364 AX-93616449 AX-93960985
##  0.01544944  0.07422969  0.07422969  0.07422969  0.07422969
```

```
P <- matrix(freqp, ncol = length(freqp), nrow = nrow(M),
    byrow = T)
ap(P)
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5]
## [1,] 0.01544944 0.07422969 0.07422969 0.07422969 0.07422969
## [2,] 0.01544944 0.07422969 0.07422969 0.07422969 0.07422969
## [3,] 0.01544944 0.07422969 0.07422969 0.07422969 0.07422969
## [4,] 0.01544944 0.07422969 0.07422969 0.07422969 0.07422969
## [5,] 0.01544944 0.07422969 0.07422969 0.07422969 0.07422969
```

```
Pbis <- P
ap(Pbis)
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5]
## [1,] 0.01544944 0.07422969 0.07422969 0.07422969 0.07422969
## [2,] 0.01544944 0.07422969 0.07422969 0.07422969 0.07422969
## [3,] 0.01544944 0.07422969 0.07422969 0.07422969 0.07422969
## [4,] 0.01544944 0.07422969 0.07422969 0.07422969 0.07422969
## [5,] 0.01544944 0.07422969 0.07422969 0.07422969 0.07422969
```

```
# I remplace al the position witout missing data by
# zero to avoid modifiying the genotyping data
ap(Pbis[c(50, 61, 340), c(1, 6, 7)])
```

```
##            [,1]      [,2]      [,3]
## [1,] 0.01544944 0.0744382 0.0744382
## [2,] 0.01544944 0.0744382 0.0744382
## [3,] 0.01544944 0.0744382 0.0744382
```

```
Pbis[!is.na(M)] <- 0
ap(Pbis[c(50, 61, 340), c(1, 6, 7)])
```

```
##            [,1]      [,2]      [,3]
## [1,] 0.01544944 0.0000000 0.0000000
## [2,] 0.00000000 0.0744382 0.0000000
## [3,] 0.00000000 0.0000000 0.0744382
```

```
# I duplicate M
Mbis <- M

# i replace the missing data by 0
ap(M[c(50, 61, 340), c(1, 6, 7)])
```

```
##            AX-93664820 AX-93664826 AX-93664827
## EUC_GM_050          NA          -1          -1
## EUC_GM_061           1          NA          -1
## EUC_GM_342           0          -1          NA
```

```
ap(Mbis[c(50, 61, 340), c(1, 6, 7)])
```

```
##            AX-93664820 AX-93664826 AX-93664827
## EUC_GM_050          NA          -1          -1
## EUC_GM_061           1          NA          -1
## EUC_GM_342           0          -1          NA
```

```
Mbis[is.na(M)] <- 0

ap(Mbis[c(50, 61, 340), c(1, 6, 7)])
```

```
##            AX-93664820 AX-93664826 AX-93664827
## EUC_GM_050           0          -1          -1
## EUC_GM_061           1           0          -1
## EUC_GM_342           0          -1           0
```

```
# I sum the two matrice in that way the allelic
# frequency will replace the missing data
M <- Mbis + Pbis

ap(M[c(50, 61, 340), c(1, 6, 7)])
```

```
##            AX-93664820 AX-93664826 AX-93664827
## EUC_GM_050  0.01544944  -1.0000000  -1.0000000
## EUC_GM_061  1.00000000   0.0744382  -1.0000000
## EUC_GM_342  0.00000000  -1.0000000   0.0744382
```

```
# I check is there is missing data
sum(is.na(M))
```

```
## [1] 0
```

```
# I remove the duplicated matrix
rm(Mbis, Pbis)
```

Now that the missing data have been replaced, the analyses can continue.

Visualization

Marker density along the genome

Observation of the distribution of markers along the chromosomes. `99.2041376` % of the markers are on the chromosomes, the rest on the scaffolds.

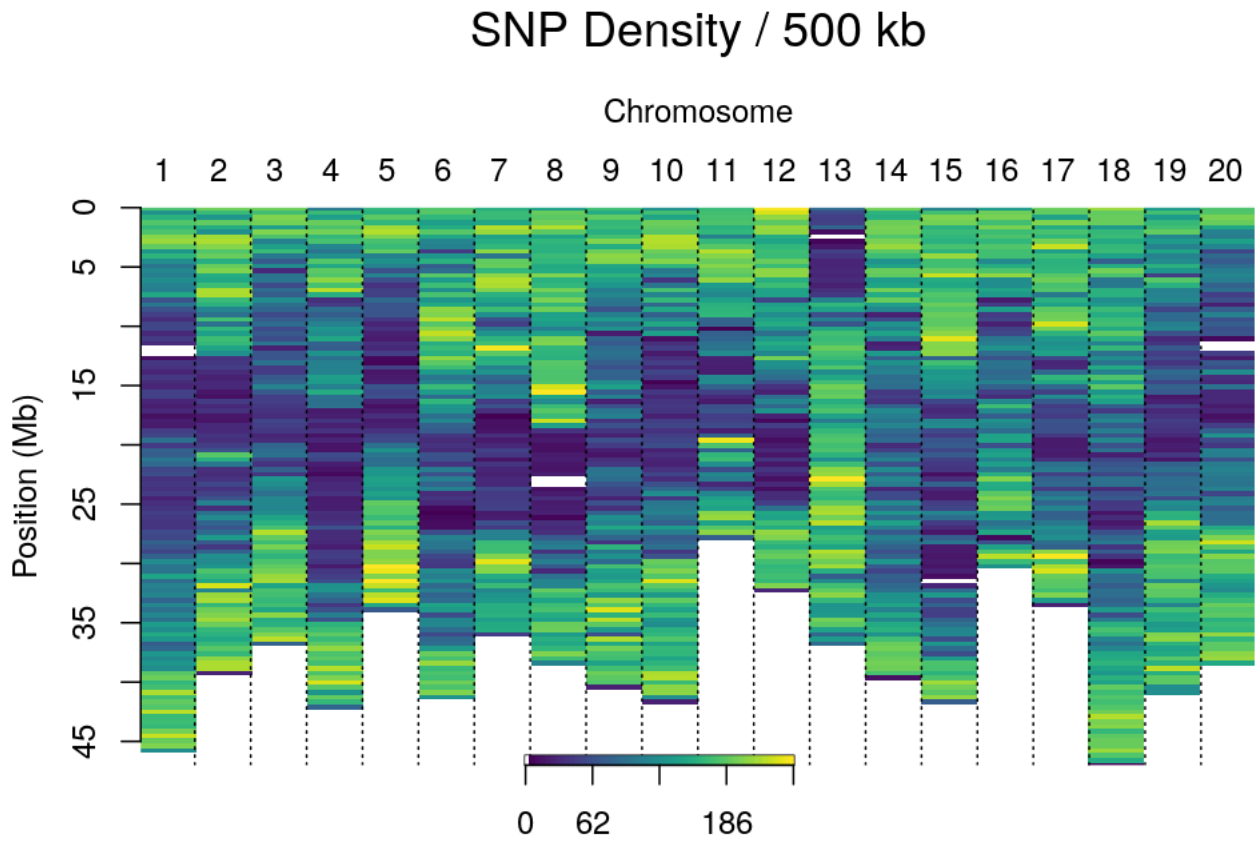
When we have the physical position of the markers, it can be interesting to look at their distribution. Are there areas of the genome that are absent or less well covered by genotyping?

We calculated the number of markers over a distance of 500 Kb. This density of markers is variable along and between chromosomes but the whole genome is covered except for five zones which are certainly centromeres. The figure below shows the marker density along the chromosomes. The more the colour goes towards purple, the lower the marker density and the more the colour is yellow, the higher the marker density. The colour white represents the absence of markers.

```
nb_chr <- length(grep("Gm", unique(Map$CHROM)))  # Number of chromosomes
nbscaf <- length(grep("scaf", unique(Map$CHROM)))  #Number of scaffold

chr_list <- sort(unique(Map$CHROM))

ma_carteall <- list()
for (i in 1:nb_chr) {
    ma_carteall[[i]] <- subset(Map, CHROM == unique(Map$CHROM)[i])$POS
}

Chrall <- list()
```

```
plot_densall <- list()
for (i in 1:length(chr_list)) {
    Chrall[[i]] <- subset(Map, CHROM == chr_list[i])[,
        c("ID", "CHROM", "POS")]
    Chrall[[i]]$"500kb" <- as.numeric(Chrall[[i]]$POS)/5e+05
    Chrall[[i]]$Mb <- as.numeric(Chrall[[i]]$POS)/1e+06
    plot_densall[[i]] <- list(`500kb` = tabulate(Chrall[[i]]$"500kb"),
        Mb = tabulate(Chrall[[i]]$Mb))
}

kb500 <- c()
for (i in 1:nb_chr) {
    kb500 <- c(kb500, length(tabulate(Chrall[[i]]$"500kb")))
}

kb1000 <- c()
for (i in 1:nb_chr) {
    kb1000 <- c(kb1000, length(tabulate(Chrall[[i]]$Mb)))
}


# heatMap density in matrix

mat_chr <- list(`500kb` = matrix(NA, nrow = max(sapply(plot_densall,
    function(x) {
        length(x$"500kb")
    })), ncol = nb_chr), Mb = matrix(NA, nrow = max(sapply(plot_densall,
    function(x) {
        length(x$Mb)
    })), ncol = nb_chr))

for (i in 1:nb_chr) {
    mat_chr$"500kb"[, i] <- c(plot_densall[[i]]$"500kb",
        rep(NA, (nrow(mat_chr$"500kb") - length(plot_densall[[i]]$"500kb"))))
    mat_chr$Mb[, i] <- c(plot_densall[[i]]$Mb, rep(NA,
        (nrow(mat_chr$Mb) - length(plot_densall[[i]]$Mb))))
}



a <- rep(-0.028, 21)
for (i in 2:length(a)) {
    a[i] <- a[i - 1] + (1.056/nb_chr)
}


op <- par(mar = c(2, 4, 6, 2))
image(t(mat_chr$"500kb"), ylim = c(1, 0), col = c(rgb(1,
    1, 1), viridis(100)), xaxt = "n", yaxt = "n", bty = "n")
abline(v = a, lty = 3)
axis(side = 2, at = 5 * (0:9)/47, labels = 5 * (0:9))
mtext("Position (Mb)", side = 2, line = 2.5, at = 0.5)
mtext(1:nb_chr, side = 3, line = 0.5, at = a[2:length(a)] -
```

```
    0.03)
mtext("Chromosome", side = 3, line = 2, at = 0.5)
mtext("SNP Density / 500 kb", side = 3, line = 4, at = 0.5,
    cex = 1.5)
par(fig = c(0.4, 0.6, 0, 0.1), mar = c(2, 0, 0, 0),
    new = TRUE)
image(matrix(c(1:(max(mat_chr$"500kb", na.rm = T))),
    max(mat_chr$"500kb", na.rm = T), 1), col = c(rgb(1,
    1, 1), viridis(100)), xaxt = "n", yaxt = "n")
axis(side = 1, at = (0:4)/4, cex.axis = 1, labels = round((0:4)/4 *
    (max(mat_chr$"500kb", na.rm = T) - 1)))
```




```
rm(Chrall, ma_carteall, mat_chr, op, plot_densall,
    a, chr_list, i, kb1000, kb500, nb_chr, nbscaf)
dev.off()
```

```
## null device
##           1
```

```
ggplot(Map)
```

Linkage disequilibrium

Another interesting parameter to look at is the linkage disequilibrium between markers. The Linkage disequilibrium is the Non random association between alleles from different (linked) loci. A quick way to do

this is to calculate a partial correlation squared.

When we have the physical (or genetic map) we can represent it according to the distance as here with chromosome 1 or only its distribution.

```
# Calculate a partial correlation squared.
LD.1 = pcor.shrink(M[, colnames(M) %in% Map$ID[Map$CHROM ==
    "Gm01"]])^2
```

```
## Estimating optimal shrinkage intensity lambda (correlation matrix): 0.1058
```

```
ap(LD.1)
```

```
##              AX-93664820  AX-93912564  AX-93616364  AX-93616449  AX-93960985
## AX-93664820 1.000000e+00 2.189614e-07 2.189614e-07 2.189614e-07 2.189614e-07
## AX-93912564 2.189614e-07 1.000000e+00 2.122623e-04 2.122623e-04 2.122623e-04
## AX-93616364 2.189614e-07 2.122623e-04 1.000000e+00 2.122623e-04 2.122623e-04
## AX-93616449 2.189614e-07 2.122623e-04 2.122623e-04 1.000000e+00 2.122623e-04
## AX-93960985 2.189614e-07 2.122623e-04 2.122623e-04 2.122623e-04 1.000000e+00
```

```
class(LD.1) = "matrix"  # Change the class object
LD.1 = as.data.frame(LD.1, stringsAsFactors = F)  # change the matrix into dataframe

LD.1[lower.tri(LD.1, diag = T)] = NA
# We keep only the upper triangle since the matrix
# is symetrical
ap(LD.1)
```

```
##             AX-93664820  AX-93912564  AX-93616364  AX-93616449  AX-93960985
## AX-93664820          NA 2.189614e-07 2.189614e-07 2.189614e-07 2.189614e-07
## AX-93912564          NA           NA 2.122623e-04 2.122623e-04 2.122623e-04
## AX-93616364          NA           NA           NA 2.122623e-04 2.122623e-04
## AX-93616449          NA           NA           NA           NA 2.122623e-04
## AX-93960985          NA           NA           NA           NA           NA
```

```
LD.1$SNP1 = rownames(LD.1)

# Switches from a matrix to a three-column table
LD.tab = gather(data = LD.1, "SNP2", "Value", -SNP1)
ap(LD.tab)
```

```
##          SNP1        SNP2 Value
## 1 AX-93664820 AX-93664820    NA
## 2 AX-93912564 AX-93664820    NA
## 3 AX-93616364 AX-93664820    NA
## 4 AX-93616449 AX-93664820    NA
## 5 AX-93960985 AX-93664820    NA
```

```
# I take out the lines with NA
LD.tab = LD.tab[!is.na(LD.tab$Value), ]

# calcul of the physical distance between marqueurs
dist.1 = as.matrix(dist(Map$POS[Map$ID %in% colnames(LD.1)],
    upper = T, diag = T))
dist.1[lower.tri(dist.1, diag = T)] = NA
dist.1 = as.data.frame(dist.1)

colnames(dist.1) = rownames(dist.1) = rownames(LD.1)
```

```
dist.1$SNP1 = rownames(dist.1)

dist.tab = gather(data = dist.1, "SNP2", "Value", -SNP1)
ap(dist.tab)
```

```
##          SNP1        SNP2 Value
## 1 AX-93664820 AX-93664820    NA
## 2 AX-93912564 AX-93664820    NA
## 3 AX-93616364 AX-93664820    NA
## 4 AX-93616449 AX-93664820    NA
## 5 AX-93960985 AX-93664820    NA
```

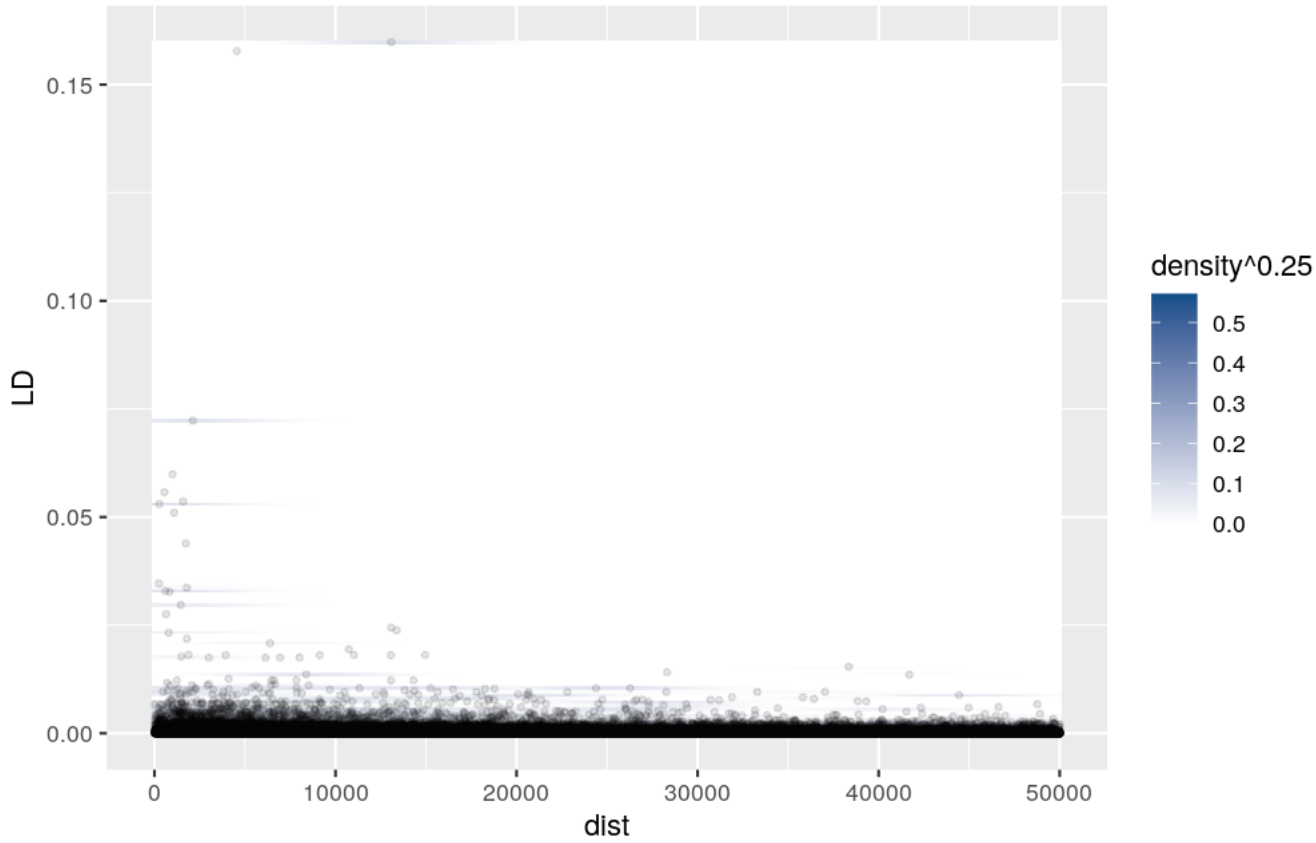
```
dist.tab = dist.tab[!is.na(dist.tab$Value), ]

# Checking the order of the lines to be able to
# concatenate the data between them
all(LD.tab$SNP1 == dist.tab$SNP1)  # should be TRUE
```

```
## [1] TRUE
```

```
all(LD.tab$SNP2 == dist.tab$SNP2)  # should be TRUE
```

```
## [1] TRUE
```

```
dist.tab$LD = LD.tab$Value

colnames(dist.tab) = c("SNP1", "SNP2", "dist", "LD")

ggplot(data = dist.tab[dist.tab$dist < 50000, ], aes(dist,
    LD)) + stat_density2d(aes(fill = ..density..^0.25),
    geom = "tile", contour = FALSE, n = 200) + scale_fill_continuous(low = "white",
    high = "dodgerblue4") + geom_point(alpha = 0.1,
    shape = 20)
```



Here, we can see that when 2 markers are separated by 15 000 bp, the LD is statistically absent.

Genetic structure

Aamir and Hilde, kindly gave me the results of their structural analysis. If a structure in the population is observed. It is always interesting to represent it graphically. It is also possible to search for it via other R packages. The structure can be used later for QTL detection, for data adjustment or to optimize the creation of training and validation populations. Many methods exist for the detection of population structuring. These include STRUCTURE, ADmixture or R routines such as Discriminant Analysis of Principal. Components (DAPC) using adegenet 2.1.0.

```
head(genoet_struc)
```

```
##     Accession Subgroup       G1       G2       G3       G4       G5
## 1: EUC_GM_001       G1 0.878352 0.000001 0.031845 0.089802 0.000001
## 2: EUC_GM_002  Admixed 0.342891 0.264961 0.183506 0.208640 0.000001
## 3: EUC_GM_003       G2 0.204714 0.070633 0.536788 0.187864 0.000001
## 4: EUC_GM_004       G1 0.873444 0.055371 0.010773 0.059112 0.001300
## 5: EUC_GM_005       G4 0.084658 0.550192 0.106250 0.159061 0.099838
## 6: EUC_GM_006       G1 0.909955 0.000001 0.000001 0.090042 0.000001
```

```
# Creation of a table with the population structur attached to the genotyping data

struc = cbind(Subgroup =genoet_struc$Subgroup[match(rownames(M),
                                                    genoet_struc$Accession)],
              as.data.frame(M,stringsAsFactors = F))
```

```
ap(struc)
```

```
##            Subgroup AX-93664820 AX-93912564 AX-93616364 AX-93616449
## EUC_GM_001       G1           1          -1           1          -1
## EUC_GM_002 Admixed           1          -1           1          -1
## EUC_GM_003       G2           1          -1           1          -1
## EUC_GM_004       G1           1          -1           1          -1
## EUC_GM_005       G4           1           1          -1           1
```

```
# Principal Analysis Composante
res.PCA <- PCA(struc,quali.sup = 1,  graph = FALSE) #

fviz_pca_ind(res.PCA,
             label = "none", # hide individual labels
             habillage = struc$Subgroup, # color by groups,
             palette = c("#C1C1C1", "#414487FF", "#2A788EFF",
                         "#22A884FF", "#7AD151FF", "#FDE725FF"),
             addEllipses = TRUE,axes = c(1,2) # Concentration ellipses
)
```


```
rm(struc,res.PCA)
```

Pedigree Preparation

If all individuals are genotyped, the pedigree can be abandoned.

When all individuals are not genotyped and a pedigree is available, It is possible to combine the matrices calculated with the pedigree and markers so that no information is lost. If the pedigree is not available for some individuals, these individuals are anyway included, and considered as unrelated.

```
head(ped)
```

```
##          Name         P1         P2 Generation       Type
## 1: EUC_GM_009 EUC_GM_239       <NA>        F12 biparental
## 2: EUC_GM_050 EUC_GM_179       <NA>        F12 biparental
## 3: EUC_GM_055 EUC_GM_204       <NA>        F12 biparental
## 4: EUC_GM_056 EUC_GM_328       <NA>        F12 biparental
## 5: EUC_GM_072 EUC_GM_328       <NA>        F12 biparental
## 6: EUC_GM_097 EUC_GM_604 EUC_GM_615        F12 biparental
```

```
tail(ped)
```

```
##          Name         P1         P2 Generation       Type
## 1: EUC_GM_938 EUC_GM_092 EUC_GM_719         F5 biparental
## 2: EUC_GM_939 EUC_GM_092 EUC_GM_719         F5 biparental
## 3: EUC_GM_940 EUC_GM_092 EUC_GM_719         F5 biparental
## 4: EUC_GM_941 EUC_GM_092 EUC_GM_719         F5 biparental
## 5: EUC_GM_942 EUC_GM_092 EUC_GM_719         F5 biparental
## 6: EUC_GM_999       <NA>       <NA>       <NA> biparental
```

```
ped <- ped[ped$Name %in% phenoSoy$EUCLEGID, ]
pedmod <- data.frame(ind = na.omit(unique(c(phenoSoy$EUCLEGID,
    ped$P1, ped$P2))), P1 = NA, P2 = NA)
pedmod$P1 <- ped$P1[match(pedmod$ind, ped$Name)]
pedmod$P2 <- ped$P2[match(pedmod$ind, ped$Name)]

pedmod$indnum = 1:nrow(pedmod)
pedmod$P1num = 0
pedmod$P2num = 0
```

When we restrict the data to the phenotypes concerned by our example, we have no remaining pedigree. We can consider all non-genotyped individuals as unrelated.

It is important to keep these individuals because otherwise we won't be able to use their phenotype in the next steps and it will be a loss of information.

Relationship matrice construction

NRM matrice (Numeric Relationship Matrix)

I am using the package *nadiv* to create an additive relationship matrix from the pedigree information (see the commented line). When there is no pedigree (like in the present case), we only need a matrix full of zero with 1 in diagonal.

```
# A <- as.matrix(makeA(pedigree =
# prepPed(pedmod[,4:6])) # If there is a pedigree

A = matrix(0, nrow(pedmod), nrow(pedmod))
```

```r
colnames(A) = rownames(A) <- pedmod$ind
diag(A) = 1
ap(A)
```

```
##            EUC_GM_310 EUC_GM_347 EUC_GM_319 EUC_GM_297 EUC_GM_341
## EUC_GM_310          1          0          0          0          0
## EUC_GM_347          0          1          0          0          0
## EUC_GM_319          0          0          1          0          0
## EUC_GM_297          0          0          0          1          0
## EUC_GM_341          0          0          0          0          1
```

GRM matrice (Genomic Relationship Matrix)

There are several methods for calculating this matrix G, one of the first methods is the method of Lynch (1988) (Lynch 1988) which was later improved by Li in 1993. This latter uses a similarity index calculated for each locus, and the relatedness coefficient is calculated by adding the index across loci and dividing by the number of loci (q) as shown by the following formulas.

$$S_{ij} = \frac{I_{x_i x_j} + I_{x_i y_j} + I_{y_i x_j} + I_{y_i y_j}}{4} \tag{1}$$

$$f_{ij} = \frac{\sum_{l=1}^{q} (S_{ij})_l}{q} \tag{2}$$

$I_{x_i x_j}$ is 1 if the allele x of the individual i is the same as the allele x (or y) of the individual j and 0 if it is not. This done, $S_{ij} $ can take four values: 0, 0.25, 0.5 and 1

This method is based on the assumption that all identical alleles (IBS) are all identical by descent (IBD). IBD means that two alleles come from the same copy present in the founding population. If this assumption is false f_{ij} is overestimated. This formula can be corrected by taking into account the probability of an allele being IBS for a locus. For this, it is necessary to know founder genotypes. If not available, there are several alternative ways to apply a correction, the first corrects the kinship coefficient with the smallest coefficient of relationship between all crosses (Hayes 2008).

$$f_{ij} = \frac{\sum_{l=1}^{q} \frac{(S_{ij})_l - S_{min}}{1 - S_{min}}}{q} \tag{3}$$

Another method is to correct the matrix G by twice the allele frequency of the minor allele as in equation (VanRaden 2007,Habier 2008).

$$G = \frac{ZZ'}{2\sum p_i(1-p_i)} \tag{4}$$

The Van Raden method is the most widely used at present, whatever the species under consideration. There are many other methods whose asumptions vary, but which give very similar results.

Regarding the genomic relationship matrix, there are several calculation methods with more or less different assumptions. In my experience, they are very similar and give similar results in terms of heritability and prediction. I propose three different methods, the second one being the most used, the last one being computed by the function of a package.

Many packages offer functions to calculate the matrices of the relationships. However, you will be dependent on the methods they have chosen. Moreover, if you want to test a new method just published, it will not

necessarily be implemented in a package. In any case, it is always interesting to know what happens under the functions you are using.

The fastest way to calculate these matrices is matrix calculation. This often looks very complicated and scares a lot of people. Some basic notions :

- How many plants ? X a vect full of 1

$$\mathbf{X} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{5}$$

$$\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} = (1 \times 1) + (1 \times 1) + (1 \times 1) = 3 \tag{6}$$

- How much do they produce?

$$\mathbf{Y} = \begin{bmatrix} 158 \\ 177 \\ 125 \end{bmatrix} \mathbf{X}^T\mathbf{Y} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 158 \\ 177 \\ 125 \end{bmatrix} = (1 \times 158) + (1 \times 177) + (1 \times 125) = 460 \tag{7}$$

$$\frac{\mathbf{X}^T\mathbf{Y}}{\mathbf{X}^T\mathbf{X}} = \frac{460}{3} = 153.33 \tag{8}$$

This is basically how kinship matrices are calculated. The sum of similarities is calculated in the numerator and the denominator is a scaling parameter.

Lopèz-Cruz et al. 2015

X is the genotyping matrice and p is the matrix with the allelic frequencies

```
X <- scale(M, scale = T, center = T)

GX = tcrossprod(X)/ncol(X)
```

VanRaden 2008 & Forni 2010

```
Z <- M - (2 * P)

den = tcrossprod(Z)
GV = den/mean(diag(den))
```

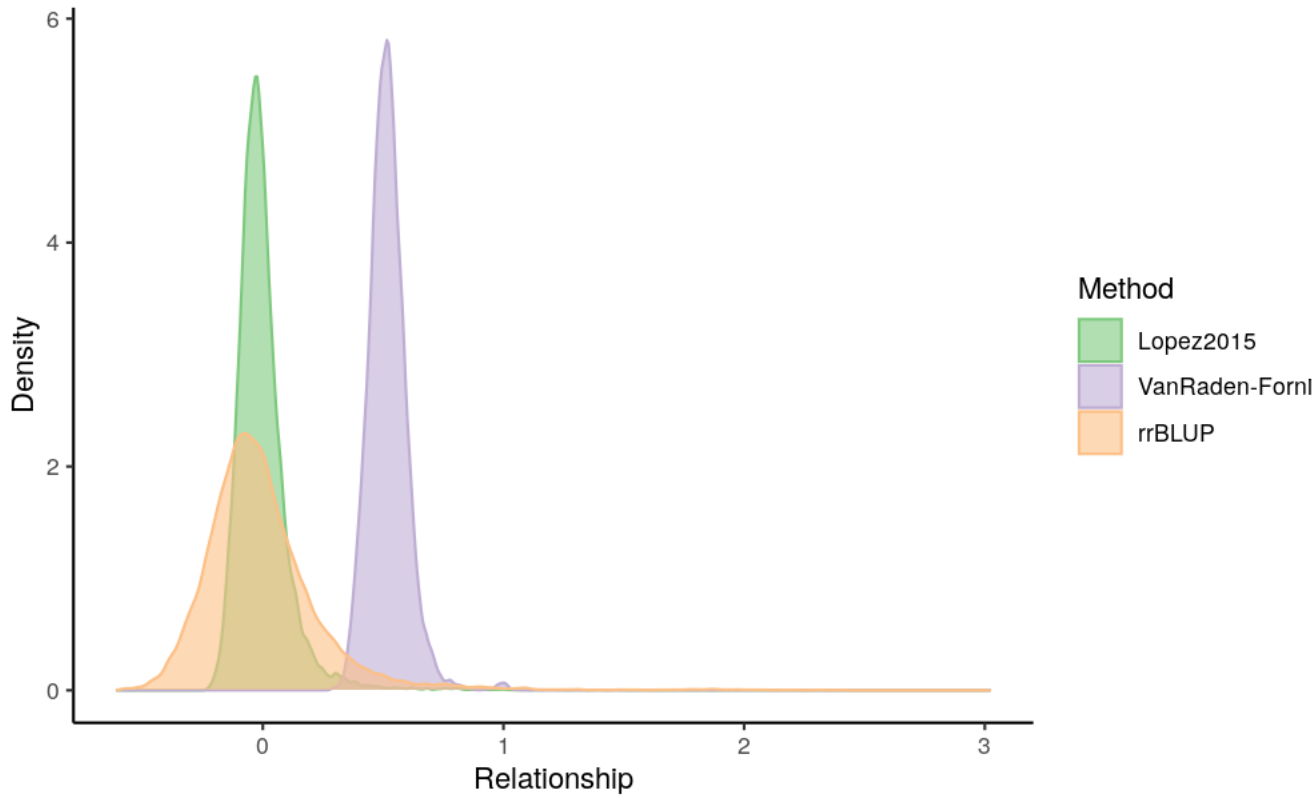
Estimated with rrblup package (Endelman et al., 2011)

```
Gr <- A.mat(M)
```

Compare plot

As shown in the following figure. Depending on the method used, the resulting matrix is different because the scalling parameter is not the same.

```
dens_GX <- data.frame(Method = "Lopez2015", x = c(GX))
dens_GV <- data.frame(Method = "VanRaden-Forni", x = c(GV))
dens_Gr <- data.frame(Method = "rrBLUP", x = c(Gr))

dens <- rbind(dens_GX, dens_GV, dens_Gr)

ggplot(dens, aes(x = x, color = Method, fill = Method)) +
    geom_density(alpha = 0.6) + scale_color_brewer(palette = "Accent") +
    scale_fill_brewer(palette = "Accent") + labs(title = "",
    x = "Relationship", y = "Density") + theme_classic()
```



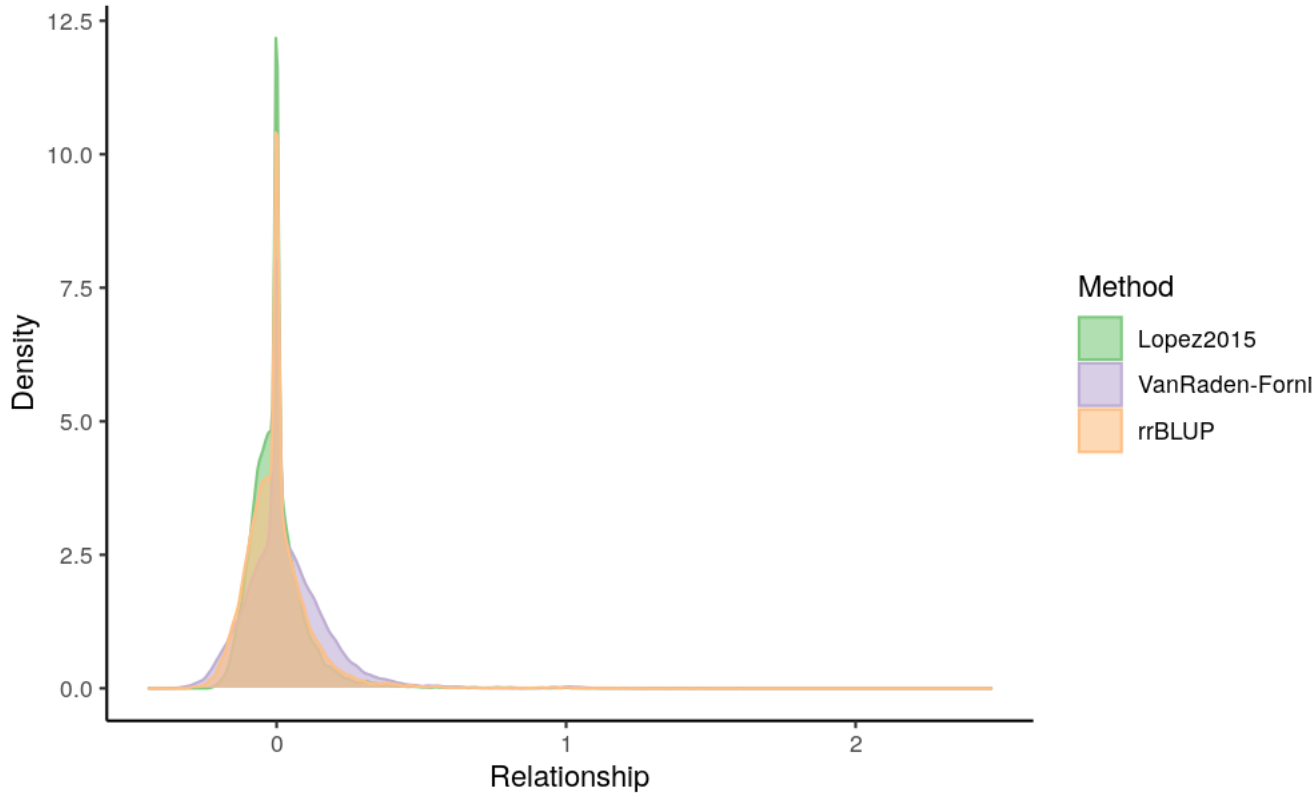

```
rm(dens_Gr, dens_GV, dens_GX)
```

We will keep these three matrices and check if there are differences in the following steps. It is not alarming to have sub-zero kinship. Depending on the method used, this could be translated as the fact that some individuals are less related than the average. If the average is zero there will be individuals with negative relationships.

Hybrid matrice

It is possible to combine any of these genomic relationship matrix with the numerical relationship matrix estimated from the pedigree to create the H matrix (for Hybrid), the difference between them is reduce.

```
HX = makeH(NRM = A, GRM = GX)
HV = makeH(NRM = A, GRM = GV)
Hr = makeH(NRM = A, GRM = Gr)

dens_HX <- data.frame(Method = "Lopez2015", x = c(HX))
dens_HV <- data.frame(Method = "VanRaden-Forni", x = c(HV))
dens_Hr <- data.frame(Method = "rrBLUP", x = c(Hr))

densH <- rbind(dens_HX, dens_HV, dens_Hr)

ggplot(densH, aes(x = x, color = Method, fill = Method)) +
    geom_density(alpha = 0.6) + scale_color_brewer(palette = "Accent") +
    scale_fill_brewer(palette = "Accent") + labs(title = "",
    x = "Relationship", y = "Density") + theme_classic()
```




```
rm(dens_Hr, dens_HV, dens_HX)
```

Heritability estimation with a global model

There are several ways to make the spatial adjustment. Most trials are submitted to spatial heterogeneities that are not completely taken into account by the block structure. It is thus important to correct the phenotypic traits by considering spatial heterogeneity . It can be done year by year or by combining the two years. This can be done by using methods that estimate effects using xy positions (or plot row column) . This is the case of the bi-splines that we will use later on.

First I will show you how to make the adjustment by separating the years and for the rest of the workshop by combining the years.

Single trait model

I will adjust year by year the environmental effect with the "EuclegID" information as a first step to represent the genetic effect. Then I will use genomic information given by the Hybrid matrix. This is for demonstration, I can use the Hybrid relationship matrix that we have estimated.

Even if the marker information is unavailable, you can take into account the genetic effect with the pedigree or with the indentity (in that case is replicates of the same genotype). To use a pedigree there is a tutorial on the breedR wiki page (follow this link : https://github.com/famuvie/breedR/wiki/Overview)

The model :

$Y_{jk} = \mu + s_k + g_j + e_{jk}$

- Y_{jk} : phenotype of the j^{th} genotype in the k^{th} spatial location (row and column)
- μ : overall mean
- s_k : the effect of the k^{th} spatial location
- g_j : the genetic effect of the j^{th} genotype
- e_{jk} : the residual

Protein Content example

2018

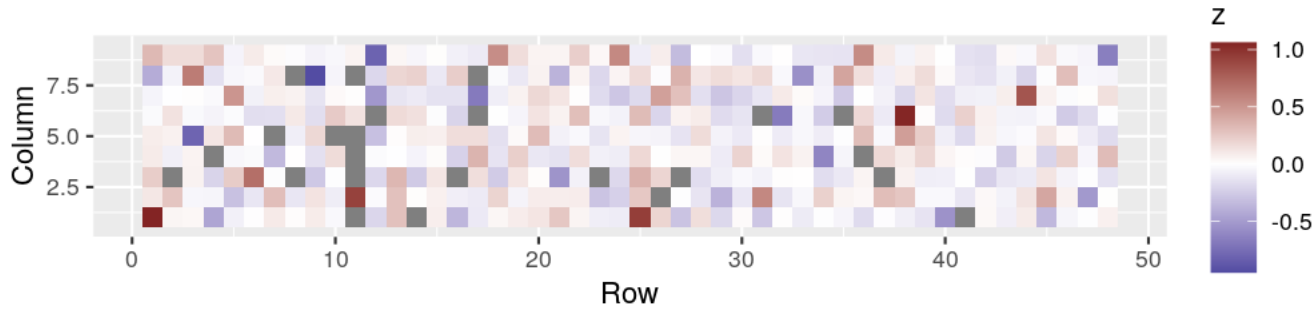
The first plot shows the model residual depending on the row and column of the field experiment. If the environmental effect is correctly taken into account we should not see any structure.

```
resA1 <- remlf90(
  fixed  =  Proteincontent  ~ 1  ,
  random = ~ EUCLEGID, #,
  spatial = list(model = 'splines',
                 coord = phenoSoy[phenoSoy$Year == 2018, c('Row', 'Column')]),
  data = phenoSoy[phenoSoy$Year == 2018,],
  method = 'em'
)
summary(resA1)
```

```
## Formula: Proteincontent ~ 0 + Intercept + EUCLEGID + spatial
##    Data: phenoSoy[phenoSoy$Year == 2018, ]
##   AIC     BIC logLik
##  1484 unknown   -739
##
## Parameters of special components:
## spatial: n.knots: 12 9
```

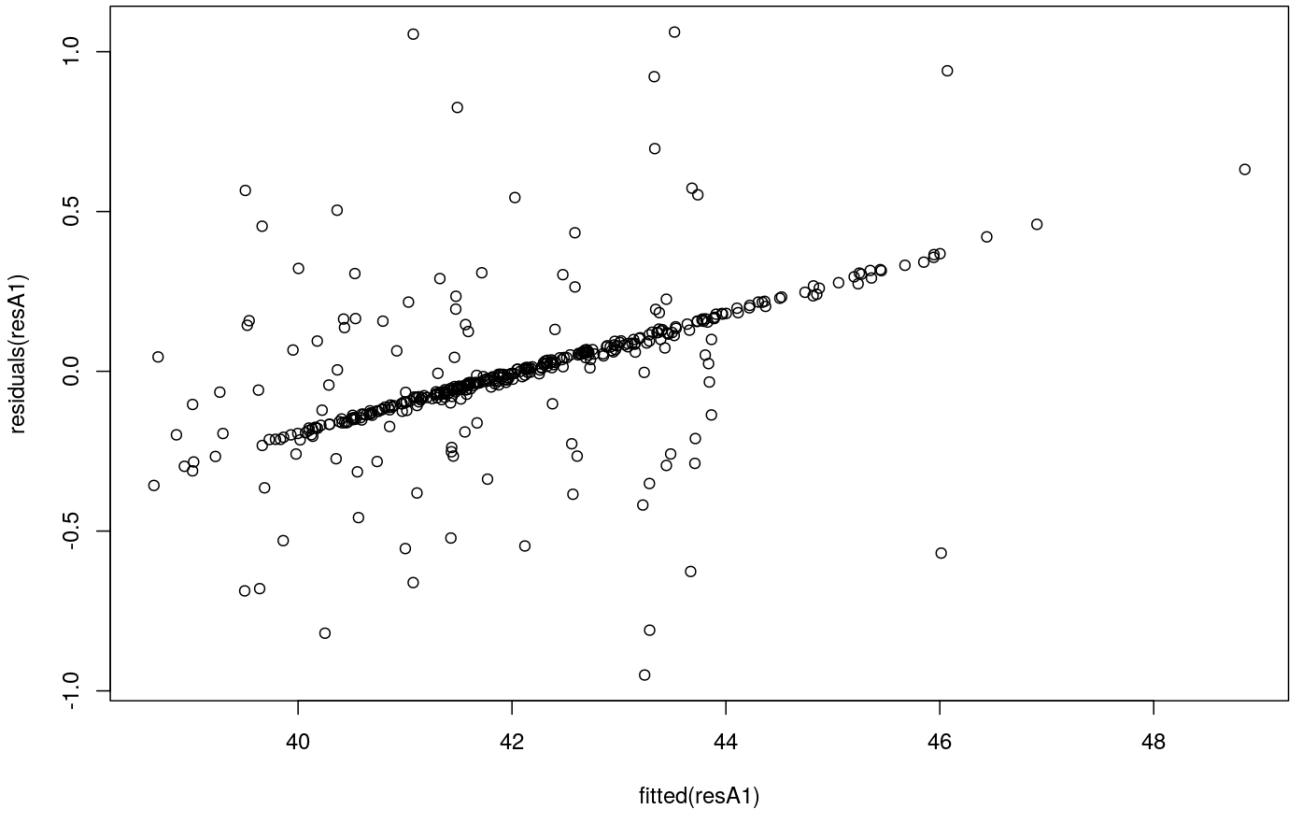
```
##
## Variance components:
##          Estimated variances
## EUCLEGID              2.47300
## spatial               0.03121
## Residual              0.23300
##
## Fixed effects:
##            value   s.e.
## Intercept 42.119 0.1307
```

```
plot(resA1, 'residuals') # Allow to check if a structure remains in the residuals
```



The second plot represents the fitted value depending on the residuals, both must be independent, it is clear here that this is not the case. An effect is not well captured by the model.

```
plot(fitted(resA1), residuals(resA1))
```

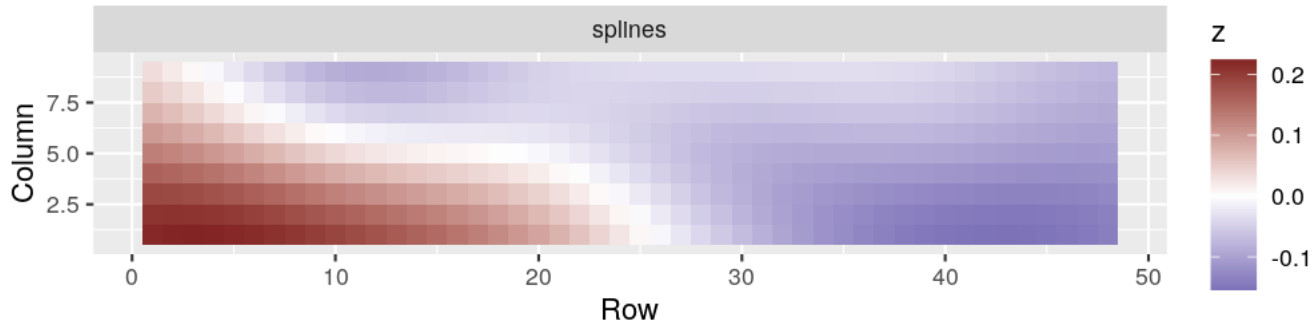



```
resA1$var["EUCLEGID", 1]/sum(resA1$var[, 1])  #heritability
```

```
## [1] 0.9034747
```

The third plot represents the field effect estimated by the model. Darker is the color stronger the field effect is. The blue represent the part of the field where the individuals produce lower protein content, and in the red part the individuals produced higher protein content.

```
plot(resA1, "spatial")  #  Represent the field differences
```



2019

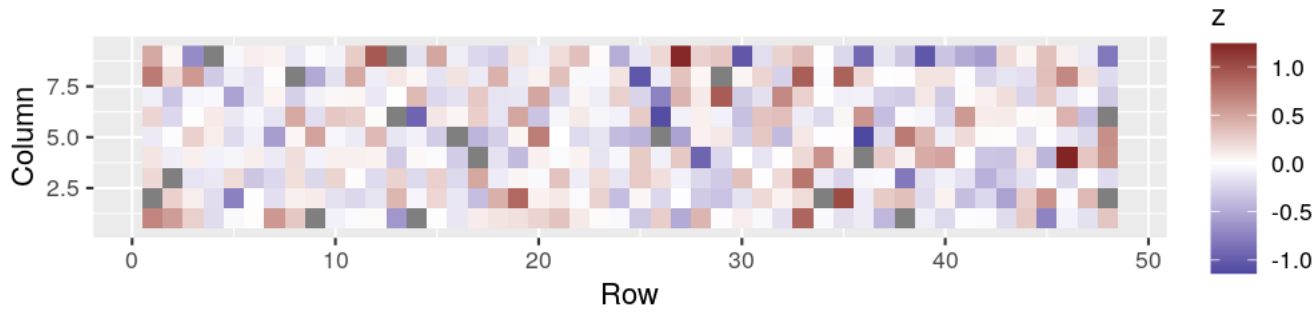
We are doing the same for the second year.

```
resA2 <- remlf90(
  fixed  =  Proteincontent  ~ 1  ,
  random = ~ EUCLEGID, #,
  spatial = list(model = 'splines',
                 coord = phenoSoy[phenoSoy$Year == 2019, c('Row', 'Column')]),
  data = phenoSoy[phenoSoy$Year == 2019,],
  method = 'em'
)
summary(resA2)
```

```
## Formula: Proteincontent ~ 0 + Intercept + EUCLEGID + spatial
##    Data: phenoSoy[phenoSoy$Year == 2019, ]
##   AIC     BIC logLik
##  1687 unknown -840.3
##
## Parameters of special components:
## spatial: n.knots: 12 9
##
## Variance components:
##          Estimated variances
## EUCLEGID              3.5090
## spatial               0.3045
```

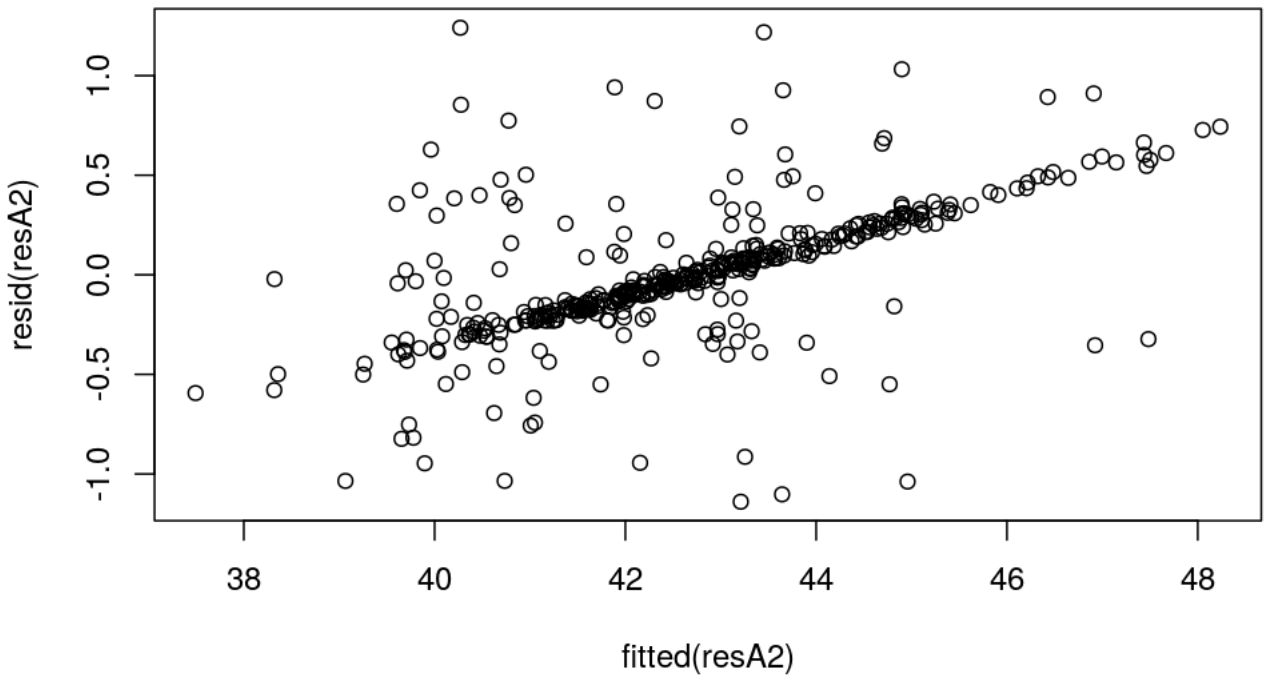
```
## Residual              0.4665
##
## Fixed effects:
##             value   s.e.
## Intercept 42.937 0.3014
```

```
plot(resA2, 'residuals') # Allow to check if a structure remains in the residuals
```



The spatial structure has disappeared

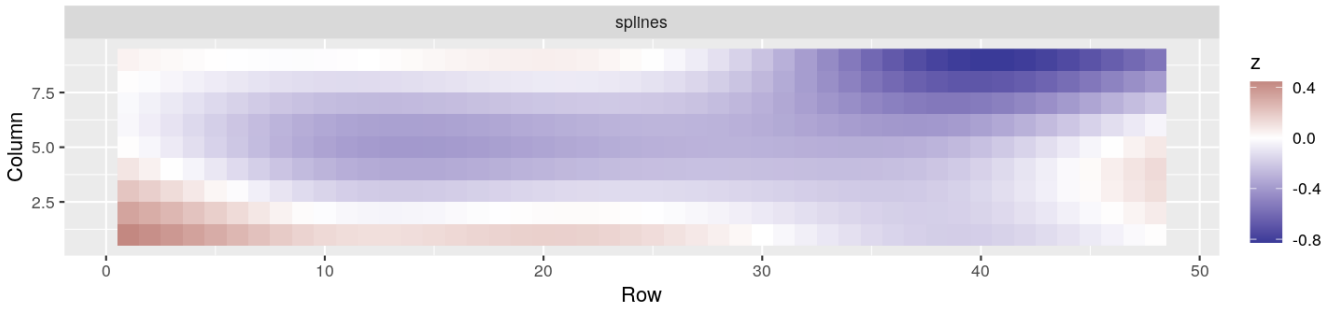
```
plot(fitted(resA2), resid(resA2))
```




```
resA2$var["EUCLEGID", 1]/sum(resA2$var[, 1])  #heritability
```

```
## [1] 0.8198598
```

```
plot(resA2, "spatial")  #  Represent the field differences
```



It is clear that the field effect is different between the two year. It may seems obvious for annual crops, but it is also true for perennial. It can even be different between cuts during the same year of growth. Now how to make a model that combines the two years. This will allow a better estimate of the genetic value of individuals because several years of phenotyping will be cumulated.

Combined

First I prepare the data, I put the year as fixed effect, so I choose to transform the variable in character. I created a table to store the results and make it easy to compare them.

The two years are different and we want the model to estimate a field effect for each year. To do this, we will create structure matrices allowing the model to distinguish the two years. In the same way, we are going to indicate the kinship between individuals present in the H matrix. We need to tell the model which line of the H matrix corresponds to the phenotype data with an incidence matrix. The incidence matrix has as many rows as the phenotyping data table and as many columns as there are columns in the H matrix. It is full of zero, we will put a 1 to link the phenotypic and genotypic data.

The model :

$Y_{ijk} = \mu + s_{kl} + g_j + a_i + e_{ijkl}$

- Y_{ijk} : phenotype of the j^{th} genotype in the i^{th} year of the k^{th} spatial location (row and column)
- μ : overall mean
- s_k : the effect of the k^{th} spatial location
- g_j : the genetic effect of the j^{th} genotype

- a_i : the year effect
- e_{ijk} : the residual

```
phenoSoy$Year = as.character(phenoSoy$Year)

comparison1 = data.frame(Method = c("Lopez2015", "VanRaden2008",
    "rrBLUP"), h2 = NA, AIC = NA)

all(c(all(colnames(HV) == colnames(HX)), all(rownames(HV) ==
    rownames(HX)), all(colnames(Hr) == colnames(HX)),
    all(rownames(Hr) == rownames(HX)), all(colnames(HV) ==
        colnames(Hr)), all(rownames(HV) == rownames(Hr))))
```

```
## [1] TRUE
```

```
# I want to use only one incidence matrix for all
# the Hybrid matrix so I check all individuals are
# in the same order.

inc.H <- matrix(0, nrow(phenoSoy), ncol = ncol(HX))
# number of line = number of lines of PhenoSoy
# number of column = number of column of HX
dim(phenoSoy)
```

```
## [1] 864  15
```

```
dim(HX)
```

```
## [1] 383 383
```

```
dim(inc.H)
```

```
## [1] 864 383
```

```
colnames(inc.H) <- colnames(HX)
rownames(inc.H) <- phenoSoy$EUCLEGID

for (i in colnames(inc.H)) {
    inc.H[which(rownames(inc.H) == i), i] <- 1
}

head(phenoSoy)  # look the EuclegID
```

```
##    Plot Row Column EntryNo   EUCLEGID                    Accessionname
## 1:    1   1      1     310 EUC_GM_310                       GL Melanie
## 2:    2   1      2     347 EUC_GM_347                         Tr\xe4ff
## 3:    3   1      3     319 EUC_GM_319                          Kasatka
## 4:    4   1      4     297 EUC_GM_297 EU_ASR_016 13-0167-B-B-046-002-B
## 5:    5   1      5     341 EUC_GM_341                         S17L2520
## 6:    6   1      6     288 EUC_GM_288                 Delitzch Nr. 36
##    Plantemergence R1 R2 R5  R8 Seedyield Proteincontent  Trial Year
## 1:              7 35 38 44 111      1668         44.581 IFVCNS 2018
## 2:              7 35 43 62  78      1291         43.629 IFVCNS 2018
## 3:              9 39 46 63  78      1912         45.052 IFVCNS 2018
## 4:              7 35 39 48 113      2274         43.380 IFVCNS 2018
## 5:              7 36 39 45 113      2490         43.233 IFVCNS 2018
## 6:              7 36 39 45  96      1967         41.911 IFVCNS 2018
```

```
ap(HX)  # check the colum names
```

```
##             EUC_GM_310   EUC_GM_347  EUC_GM_319  EUC_GM_297   EUC_GM_341
## EUC_GM_310  0.70410518 -0.033774058 -0.05676455  0.01845636  0.038903473
## EUC_GM_347 -0.03377406  0.652405433  0.02249552  0.06585859  0.004314007
## EUC_GM_319 -0.05676455  0.022495518  1.13199784 -0.06510567 -0.045598992
## EUC_GM_297  0.01845636  0.065858593 -0.06510567  0.95777909  0.015038527
## EUC_GM_341  0.03890347  0.004314007 -0.04559899  0.01503853  0.710453426
```

```
ap(inc.H)  # now the incidence matrix
```

```
##            EUC_GM_310 EUC_GM_347 EUC_GM_319 EUC_GM_297 EUC_GM_341
## EUC_GM_310          1          0          0          0          0
## EUC_GM_347          0          1          0          0          0
## EUC_GM_319          0          0          1          0          0
## EUC_GM_297          0          0          0          1          0
## EUC_GM_341          0          0          0          0          1
```

```
# Now in the case of duplicated individuals
phenoSoy[phenoSoy$EUCLEGID == "EUC_GM_001", c(1:3,
    5, 11:15)]
```

```
##     Plot Row Column   EUCLEGID  R8 Seedyield Proteincontent  Trial Year
##  1:  339  38      6 EUC_GM_001 131      3437         42.132 IFVCNS 2018
##  2:  352  40      1 EUC_GM_001 131      3090         40.448 IFVCNS 2018
##  3:  367  41      7 EUC_GM_001 125      3598         41.004 IFVCNS 2018
##  4:  380  43      2 EUC_GM_001 125      3233         40.940 IFVCNS 2018
##  5:  409  46      4 EUC_GM_001 131      3376         41.249 IFVCNS 2018
##  6:  432  48      9 EUC_GM_001 132      3672         40.415 IFVCNS 2018
##  7:  339  38      6 EUC_GM_001 141      3233         39.770 IFVCNS 2019
##  8:  352  40      1 EUC_GM_001 141      2950         39.800 IFVCNS 2019
##  9:  367  41      7 EUC_GM_001 145      3102         39.720 IFVCNS 2019
## 10:  380  43      2 EUC_GM_001 141      3006         39.570 IFVCNS 2019
## 11:  409  46      4 EUC_GM_001 141      2762         41.510 IFVCNS 2019
## 12:  432  48      9 EUC_GM_001 141      3190         38.830 IFVCNS 2019
```

```
HX[285:290, 285:290]  # check the colum names
```

```
##             EUC_GM_001   EUC_GM_004   EUC_GM_076   EUC_GM_050  EUC_GM_075
## EUC_GM_001 0.78504409  0.124498287  0.015290142  0.097864313  0.09061471
## EUC_GM_004 0.12449829  0.979607959  0.017276042 -0.003290733  0.30883603
## EUC_GM_076 0.01529014  0.017276042  1.102803036 -0.014318045  0.07306062
## EUC_GM_050 0.09786431 -0.003290733 -0.014318045  0.684801274 -0.01326808
## EUC_GM_075 0.09061471  0.308836035  0.073060624 -0.013268079  1.39853649
## EUC_GM_039 0.05850470  0.246524555 -0.008883417  0.024661069  0.63847984
##              EUC_GM_039
## EUC_GM_001  0.058504704
## EUC_GM_004  0.246524555
## EUC_GM_076 -0.008883417
## EUC_GM_050  0.024661069
## EUC_GM_075  0.638479845
## EUC_GM_039  1.665476597
```

```
inc.H[which(rownames(inc.H) == "EUC_GM_001"), 285:290]  # now the incidence matrix
```

```
##            EUC_GM_001 EUC_GM_004 EUC_GM_076 EUC_GM_050 EUC_GM_075 EUC_GM_039
## EUC_GM_001          1          0          0          0          0          0
```

```
## EUC_GM_001          1          0          0          0          0          0
## EUC_GM_001          1          0          0          0          0          0
## EUC_GM_001          1          0          0          0          0          0
## EUC_GM_001          1          0          0          0          0          0
## EUC_GM_001          1          0          0          0          0          0
## EUC_GM_001          1          0          0          0          0          0
## EUC_GM_001          1          0          0          0          0          0
## EUC_GM_001          1          0          0          0          0          0
## EUC_GM_001          1          0          0          0          0          0
## EUC_GM_001          1          0          0          0          0          0
## EUC_GM_001          1          0          0          0          0          0
```

```
# See that each line points to the corresponding
# column

# now I am doing the same thing but for the spatial
# effect of each year
sp18 <- breedR:::breedr_splines(phenoSoy[phenoSoy$Year ==
    "2018", c("Row", "Column")])
sp19 <- breedR:::breedr_splines(phenoSoy[phenoSoy$Year ==
    "2019", c("Row", "Column")])

m18 <- model.matrix(sp18)
inc.sp18 <- Matrix::Matrix(0, nrow(phenoSoy), ncol = ncol(m18))
inc.sp18[phenoSoy$Year == "2018", ] <- m18

m19 <- model.matrix(sp19)
inc.sp19 <- Matrix::Matrix(0, nrow(phenoSoy), ncol = ncol(m19))
inc.sp19[phenoSoy$Year == "2019", ] <- m19
```

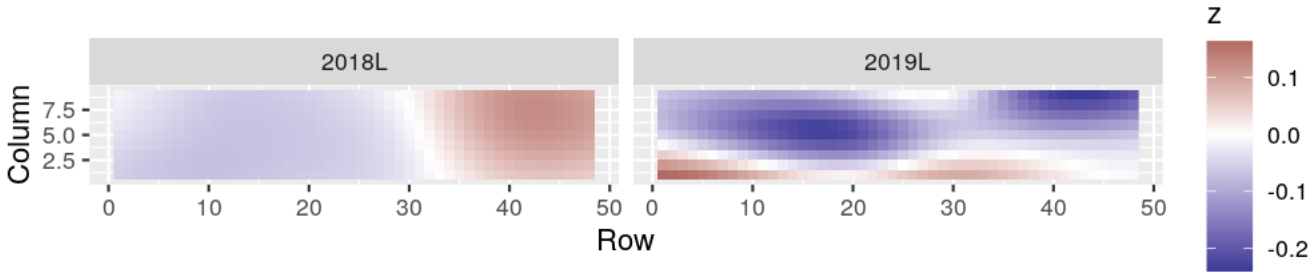
The expression 'pkg::name' returns the value of the exported variable 'name' in package 'pkg' if the package has a name space. The expression 'pkg:::name' returns the value of the internal variable 'name' in package 'pkg' if the package has a name space.

The pattern of the micro-environmental effect obtained by combining the two years is very similar to that obtained separately but is more clearly defined.

protein content Two years combined model with the matrix HX (Lopèz-Cruz)

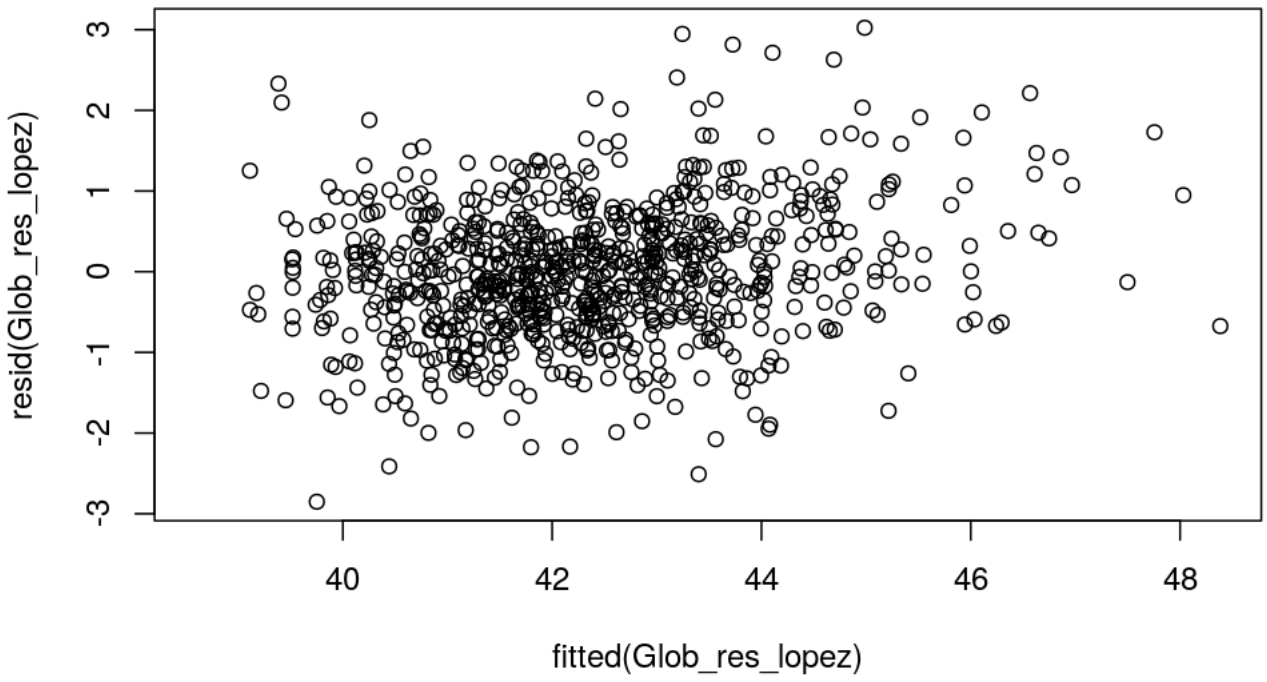
```
Glob_res_lopez <- remlf90(
  fixed  =  Proteincontent ~ 1 + Year,
  # random = ~  1, # If you need random effects
  generic = list(genetic = list(inc.H,HX),
                 sp18 = list(inc.sp18,
                             breedR:::get_structure(sp18)),
                 sp19 = list(inc.sp19,
                             breedR:::get_structure(sp19))
  ),
  data = phenoSoy,
  method = 'em'
)

psp18L <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp18),
```

```
    z = as.vector(m18 %*% ranef(Glob_res_lopez)$sp18)
  )
)
psp19L <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp19),
    z = as.vector(m19 %*% ranef(Glob_res_lopez)$sp19)
  )
)

compare.plots(list("2018L" = psp18L,
                   "2019L" = psp19L))
```



Whether combining the data or using the matching matrix, the data generated by the model and the residuals are independent.

```
plot(fitted(Glob_res_lopez), resid(Glob_res_lopez))
```




```
summary(Glob_res_lopez)
```

```
## Formula: Proteincontent ~ 0 + Intercept + Year
##    Data: phenoSoy
##   AIC     BIC logLik
##  2876 unknown  -1434
##
## Parameters of special components:
##
##
## Variance components:
##                 Estimated variances
## generic_genetic             2.46900
## sp18                        0.01584
## sp19                        0.06144
## Residual                    0.95260
##
## Fixed effects:
##              value   s.e.
## Intercept 42.85505 0.1648
## Year.2018 -0.69749 0.1643
## Year.2019  0.00000 0.0000
```

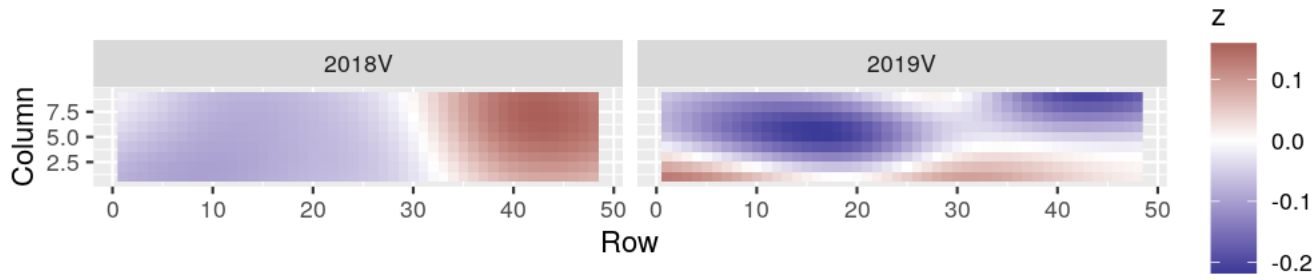
```
Glob_res_lopez$var["generic_genetic", 1]/sum(Glob_res_lopez$var[,
    1])  #heritability
```

```
## [1] 0.7056544
```

```
comparison1$h2[comparison1$Method == "Lopez2015"] = Glob_res_lopez$var["generic_genetic",
    1]/sum(Glob_res_lopez$var[, 1])
comparison1$AIC[comparison1$Method == "Lopez2015"] = Glob_res_lopez$fit$AIC
```

protein content Two years combined model with the matrix HV (VanRaden)

```
Glob_res_VanRaden <- remlf90(
  fixed  =  Proteincontent ~ 1 + Year,
  # random = ~  1, # If you need random effects
  generic = list(genetic = list(inc.H,HV),
                 sp18 = list(inc.sp18,
                             breedR:::get_structure(sp18)),
                 sp19 = list(inc.sp19,
                             breedR:::get_structure(sp19))
  ),
  data = phenoSoy,
  method = 'em'
)

psp18V <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp18),
    z = as.vector(m18 %*% ranef(Glob_res_VanRaden)$sp18)
  )
)
psp19V <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp19),
    z = as.vector(m19 %*% ranef(Glob_res_VanRaden)$sp19)
  )
)

compare.plots(list("2018V" = psp18V,
                   "2019V" = psp19V))
```




```
summary(Glob_res_VanRaden)
```

```
## Formula: Proteincontent ~ 0 + Intercept + Year
##    Data: phenoSoy
##   AIC     BIC logLik
##  2880 unknown  -1436
##
## Parameters of special components:
##
##
## Variance components:
##                 Estimated variances
## generic_genetic              2.5320
## sp18                         0.0210
## sp19                         0.0527
## Residual                     0.9658
##
## Fixed effects:
##              value   s.e.
## Intercept 43.09632 0.1379
## Year.2018 -0.68757 0.1618
## Year.2019  0.00000 0.0000
```

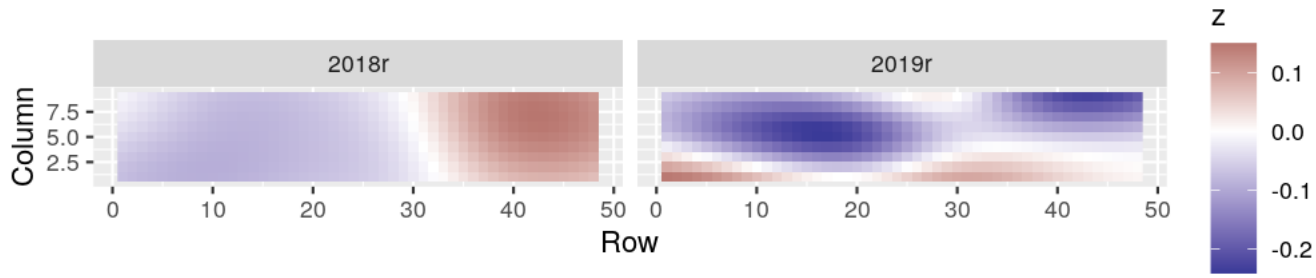
```
Glob_res_VanRaden$var["generic_genetic",1]/sum(Glob_res_VanRaden$var[,1])#heritability
```

```
## [1] 0.7089458
```

```
comparison1$h2[comparison1$Method == "VanRaden2008"] =
  Glob_res_VanRaden$var["generic_genetic",1]/sum(Glob_res_VanRaden$var[,1])
comparison1$AIC[comparison1$Method == "VanRaden2008"] =
  Glob_res_VanRaden$fit$AIC
```

protein content Two years combined model with the matrix Hr (rrBLUP)

```
Glob_res_rrBLUP <- remlf90(
  fixed  =  Proteincontent ~ 1 + Year,
  # random = ~  1, # If you need random effects
  generic = list(genetic = list(inc.H,Hr),
                 sp18 = list(inc.sp18,
                             breedR:::get_structure(sp18)),
                 sp19 = list(inc.sp19,
                             breedR:::get_structure(sp19))
  ),
  data = phenoSoy,
  method = 'em'
)

psp18r <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp18),
    z = as.vector(m18 %*% ranef(Glob_res_rrBLUP)$sp18)
  )
)
psp19r <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp19),
    z = as.vector(m19 %*% ranef(Glob_res_rrBLUP)$sp19)
  )
)

compare.plots(list("2018r" = psp18r,
                   "2019r" = psp19r))
```




```
summary(Glob_res_rrBLUP)
```

```
## Formula: Proteincontent ~ 0 + Intercept + Year
##    Data: phenoSoy
##   AIC     BIC logLik
##  2881 unknown  -1436
##
## Parameters of special components:
##
##
## Variance components:
##                 Estimated variances
## generic_genetic             2.55700
## sp18                        0.01971
## sp19                        0.06089
## Residual                    0.96410
##
## Fixed effects:
##             value   s.e.
## Intercept 42.8522 0.1652
## Year.2018 -0.6972 0.1673
## Year.2019  0.0000 0.0000
```

```
Glob_res_rrBLUP$var["generic_genetic",1]/sum(Glob_res_rrBLUP$var[,1])#heritability
```

```
## [1] 0.7099425
```

```
comparison1$h2[comparison1$Method == "rrBLUP"] =
  Glob_res_rrBLUP$var["generic_genetic",1]/sum(Glob_res_rrBLUP$var[,1])
comparison1$AIC[comparison1$Method == "rrBLUP"] =
  Glob_res_rrBLUP$fit$AIC
```

Results comparaison

Whatever the matrix to use the pattern is very close. Similarly, the heritability and the AIC parameter are equivalent.

```
comparison1
```

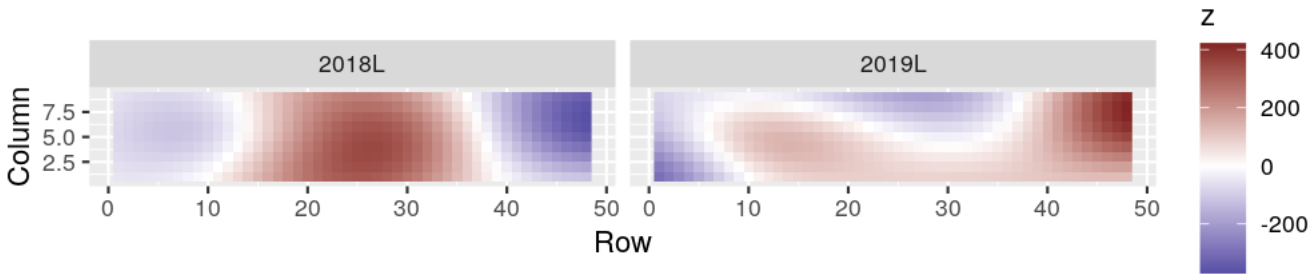
```
##         Method        h2      AIC
## 1    Lopez2015 0.7056544 2875.645
## 2 VanRaden2008 0.7089458 2879.898
## 3       rrBLUP 0.7099425 2880.666
```

Now the same with the Seedyield

Seedyield Two years combined model with the matrix HX (Lopèz-Cruz)

```
comparison2 = data.frame(Method = c("Lopez2015", "VanRaden2008",
    "rrBLUP"), h2 = NA, AIC = NA)
```

```
Glob_res_lopez <- remlf90(
  fixed  =  Seedyield ~ 1 + Year,
  # random = ~  1, # If you need random effects
  generic = list(genetic = list(inc.H,HX),
                 sp18 = list(inc.sp18,
                             breedR:::get_structure(sp18)),
                 sp19 = list(inc.sp19,
                             breedR:::get_structure(sp19))
  ),
  data = phenoSoy,
  method = 'em'
)

psp18L <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp18),
    z = as.vector(m18 %*% ranef(Glob_res_lopez)$sp18)
  )
)
psp19L <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp19),
    z = as.vector(m19 %*% ranef(Glob_res_lopez)$sp19)
  )
)

compare.plots(list("2018L" = psp18L,
                   "2019L" = psp19L))
```




```
summary(Glob_res_lopez)
```

```
## Formula: Seedyield ~ 0 + Intercept + Year
##    Data: phenoSoy
##    AIC     BIC logLik
##  13479 unknown  -6736
##
## Parameters of special components:
##
##
## Variance components:
##                 Estimated variances
## generic_genetic              206200
## sp18                          44830
## sp19                          54460
## Residual                     249200
##
## Fixed effects:
##             value   s.e.
## Intercept 2140.92 123.90
## Year.2018  170.95 163.45
## Year.2019    0.00   0.00
```

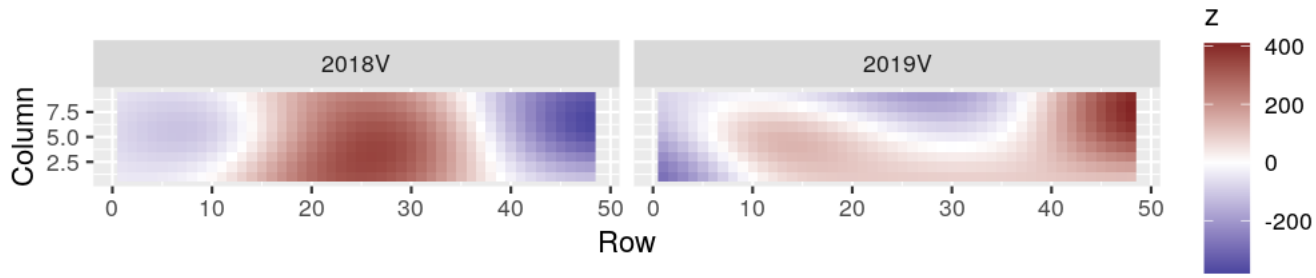
```
Glob_res_lopez$var["generic_genetic",1]/sum(Glob_res_lopez$var[,1])#heritability
```

```
## [1] 0.3717392
```

```
comparison2$h2[comparison2$Method == "Lopez2015"] =
  Glob_res_lopez$var["generic_genetic",1]/sum(Glob_res_lopez$var[,1])
comparison2$AIC[comparison2$Method == "Lopez2015"] = Glob_res_lopez$fit$AIC
```

Seedyield Two years combined model with the matrix HV (VanRaden)

```
Glob_res_VanRaden <- remlf90(
  fixed  =  Seedyield ~ 1 + Year,
  # random = ~  1, # If you need random effects
  generic = list(genetic = list(inc.H,HV),
                 sp18 = list(inc.sp18,
                             breedR:::get_structure(sp18)),
                 sp19 = list(inc.sp19,
                             breedR:::get_structure(sp19))
  ),
  data = phenoSoy,
  method = 'em'
)

psp18V <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp18),
    z = as.vector(m18 %*% ranef(Glob_res_VanRaden)$sp18)
  )
)
psp19V <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp19),
    z = as.vector(m19 %*% ranef(Glob_res_VanRaden)$sp19)
  )
)

compare.plots(list("2018V" = psp18V,
                   "2019V" = psp19V))
```




```
summary(Glob_res_VanRaden)
```

```
## Formula: Seedyield ~ 0 + Intercept + Year
##    Data: phenoSoy
##    AIC     BIC logLik
##  13476 unknown  -6734
##
## Parameters of special components:
##
##
## Variance components:
##                 Estimated variances
## generic_genetic              206800
## sp18                          45780
## sp19                          52500
## Residual                     250000
##
## Fixed effects:
##             value   s.e.
## Intercept 2036.94 120.33
## Year.2018  169.81 162.75
## Year.2019    0.00   0.00
```

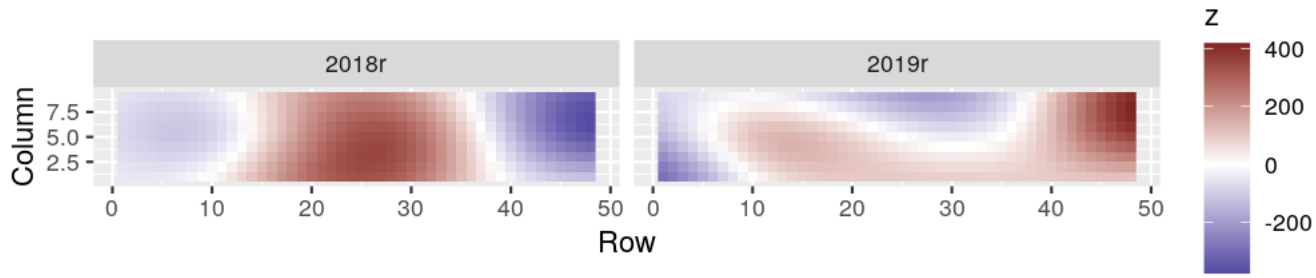
```
Glob_res_VanRaden$var["generic_genetic",1]/sum(Glob_res_VanRaden$var[,1])#heritability
```

```
## [1] 0.3725589
```

```
comparison2$h2[comparison2$Method == "VanRaden2008"] =
  Glob_res_VanRaden$var["generic_genetic",1]/sum(Glob_res_VanRaden$var[,1])
comparison2$AIC[comparison2$Method == "VanRaden2008"] =
  Glob_res_VanRaden$fit$AIC
```

Seedyield Two years combined model with the matrix Hr (rrBLUP)

```
Glob_res_rrBLUP <- remlf90(
  fixed  =  Seedyield ~ 1 + Year,
  # random = ~  1, # If you need random effects
  generic = list(genetic = list(inc.H,Hr),
                 sp18 = list(inc.sp18,
                             breedR:::get_structure(sp18)),
                 sp19 = list(inc.sp19,
                             breedR:::get_structure(sp19))
  ),
  data = phenoSoy,
  method = 'em'
)

psp18r <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp18),
    z = as.vector(m18 %*% ranef(Glob_res_rrBLUP)$sp18)
  )
)
psp19r <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp19),
    z = as.vector(m19 %*% ranef(Glob_res_rrBLUP)$sp19)
  )
)

compare.plots(list("2018r" = psp18r,
                   "2019r" = psp19r))
```




```
summary(Glob_res_rrBLUP)
```

```
## Formula: Seedyield ~ 0 + Intercept + Year
##    Data: phenoSoy
##    AIC     BIC logLik
##  13478 unknown  -6735
##
## Parameters of special components:
##
##
## Variance components:
##                 Estimated variances
## generic_genetic              211300
## sp18                          45540
## sp19                          54250
## Residual                     249800
##
## Fixed effects:
##             value   s.e.
## Intercept 2143.80 123.74
## Year.2018  170.71 163.84
## Year.2019    0.00   0.00
```

```
Glob_res_rrBLUP$var["generic_genetic",1]/sum(Glob_res_rrBLUP$var[,1]) #heritability
```

```
## [1] 0.3767227
```

```
comparison2$h2[comparison2$Method == "rrBLUP"] =
  Glob_res_rrBLUP$var["generic_genetic",1]/sum(Glob_res_rrBLUP$var[,1])
comparison2$AIC[comparison2$Method == "rrBLUP"] =
  Glob_res_rrBLUP$fit$AIC
```

Results comparaison

For Seedyield the conclusions are the same as for proteincontent. The three matrices give similar results.

```
comparison2
```

```
##         Method        h2      AIC
## 1    Lopez2015 0.3717392 13479.42
## 2 VanRaden2008 0.3725589 13475.82
## 3       rrBLUP 0.3767227 13478.36
```

The model presented here remains simple, it is possible to complicate it with multi-environment if you wish or with more years. It is also possible to adjust 2 or more traits at the same time via a multi-trait model.

Estimation of genetic correlation with a multiple-trait model

With breedR the multi-trait model is possible and allows to estimate genetic effects per trait, an environmental effect per trait as well. It also allows to estimate the genetic correlation between different traits. This is what we will see below. Here we will estimate the genetic correlation between Seedyield, Proteincontent and maturity (R8).

```
MT_Glob_res_lopez <- remlf90(
  fixed  =  cbind(Proteincontent,Seedyield,R8) ~ 1 + Year,
  # random = ~  1, # If you need random effects
  generic = list(genetic = list(inc.H,HX),
                 sp18 = list(inc.sp18,
                             breedR:::get_structure(sp18)),
                 sp19 = list(inc.sp19,
                             breedR:::get_structure(sp19))
  ),
  data = phenoSoy,
  method = 'em'
)

summary(MT_Glob_res_lopez)
```

```
## Formula: cbind(Proteincontent, Seedyield, R8) ~ 0 + Intercept + Year
##    Data: phenoSoy
##    AIC     BIC logLik
##  18892 unknown  -9422
##
## Parameters of special components:
##
##
## Variance components:
## $generic_genetic
##                Proteincontent Seedyield       R8
## Proteincontent          2.531    -212.6   -2.849
## Seedyield            -212.600  224600.0 2827.000
```

```
## R8                     -2.849    2827.0   70.500
##
## $sp18
##                Proteincontent Seedyield       R8
## Proteincontent        0.07897    -53.97    1.177
## Seedyield           -53.97000  62610.00 -531.000
## R8                    1.17700   -531.00   21.900
##
## $sp19
##                Proteincontent Seedyield     R8
## Proteincontent        0.06071    -35.43  0.234
## Seedyield           -35.43000  35750.00 54.790
## R8                    0.23400     54.79  9.675
##
## $Residual
##                Proteincontent Seedyield      R8
## Proteincontent         0.9541    -42.49   1.935
## Seedyield             -42.4900 247200.00 247.700
## R8                      1.9350    247.70  27.410
##
##
## Fixed effects:
##                                value     s.e.
## Intercept.Proteincontent   42.86681   0.1615
## Intercept.Seedyield      2136.19016 103.5175
## Intercept.R8              131.47960   1.6462
## Year.Proteincontent.2018   -0.72596   0.1999
## Year.Proteincontent.2019    0.00000   0.0000
## Year.Seedyield.2018       193.18527 161.6745
## Year.Seedyield.2019         0.00000   0.0000
## Year.R8.2018              -20.50778   2.7654
## Year.R8.2019                0.00000   0.0000
```

```
#heritability
heritabilities = diag(MT_Glob_res_lopez$var$generic_genetic)/(diag(MT_Glob_res_lopez$var$sp18)+
                                                              diag(MT_Glob_res_lopez$var$sp19)+
                                                              diag(MT_Glob_res_lopez$var$generic_genetic)+
                                                              diag(MT_Glob_res_lopez$var$Residual))

heritabilities
```

```
## Proteincontent      Seedyield             R8
##      0.6982493      0.3939245      0.5444646
```

```
#genetic correlation
cov2cor(MT_Glob_res_lopez$var$generic_genetic)
```

```
##                Proteincontent  Seedyield         R8
## Proteincontent      1.0000000 -0.2819761 -0.2132808
## Seedyield          -0.2819761  1.0000000  0.7104383
## R8                 -0.2132808  0.7104383  1.0000000
```

The heritability obtained with the multi-trait model is equivalent for Proteincontent but higher for Seedyield. Genetic correlation analysis reveals that maturity is strongly correlated with Seedyield. It may be of interest to correct Seedyield by maturity. Proteincontent is negatively correlated with maturity and Seedyield.

Phenotypic adjustment

To adjust the phenotype, the unwanted effects of the observed phenotype are removed. The genetic part and the residual are retained. We could extract only the genetic part, in this case we would have blups, which are very close to the EBV estimated with the pedigree. In theory it contains only the additive part transmitted by the parents. The other effects like dominance and epistasis are found in the residual. This is why I prefer to keep the complete phenotype from which I remove the micro-environmental part to avoid losing any genetic information.

To have only one phenotype per genotype we calculate the genotypic mean. It is normal that the phenotype is not identical between the two years even if the year effect is removed, since the heritability of the trait is not 1.

The model :

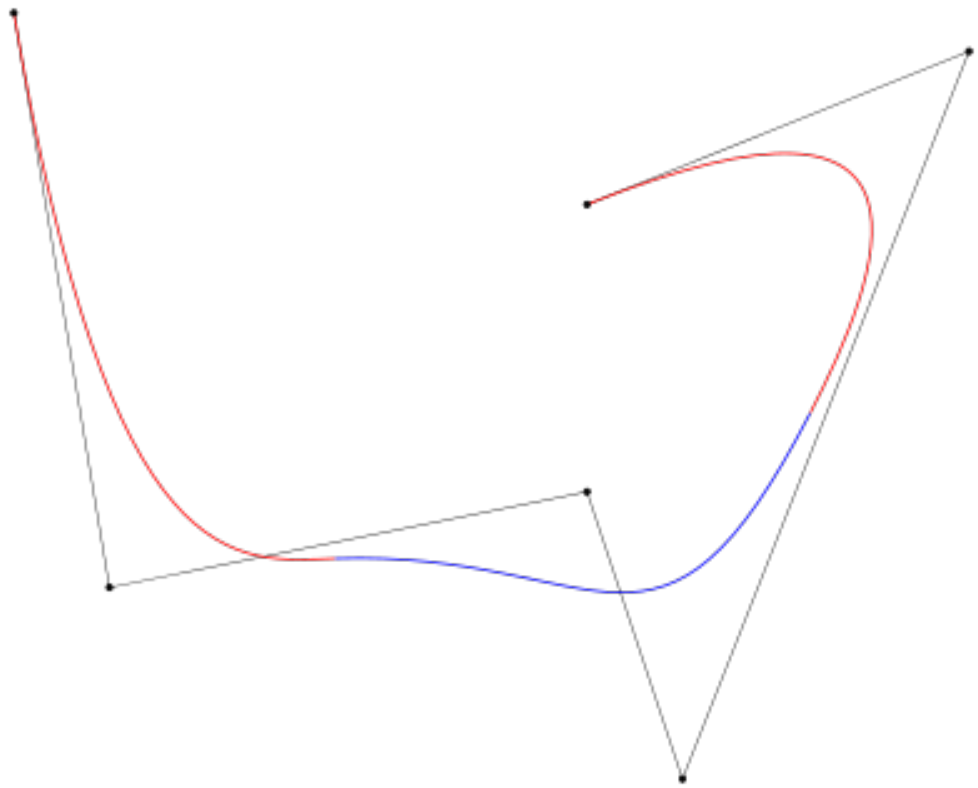
$Y_{ijk} = \mu + s_{kl} + g_j + a_i + m_{ijk} + e_{ijk}$

- Y_{ijk} = phenotype of the j^{th} genotype in the i^{th} year of the k^{th} spatial location (row and column)
- μ : overall mean
- s_k : the effect of the k^{th} spatial location
- g_j : the genetic effect of the j^{th} genotype
- m_{ijk} : the maturity of the j^{th} genotype in the i^{th} year of the k^{th} spatial location (row and column)
- e_{ijk} : the residual

The year is defined as fixed effect, the rest are random effects.

Bi-splines

- A B-spline is a linear combination of positive splines
- B-splines are the generalisation of the Bézier curves
- The shape of the basic functions is determined by the position of the nodes.
- The curve is within the convex envelope of the control points.



Seedyield

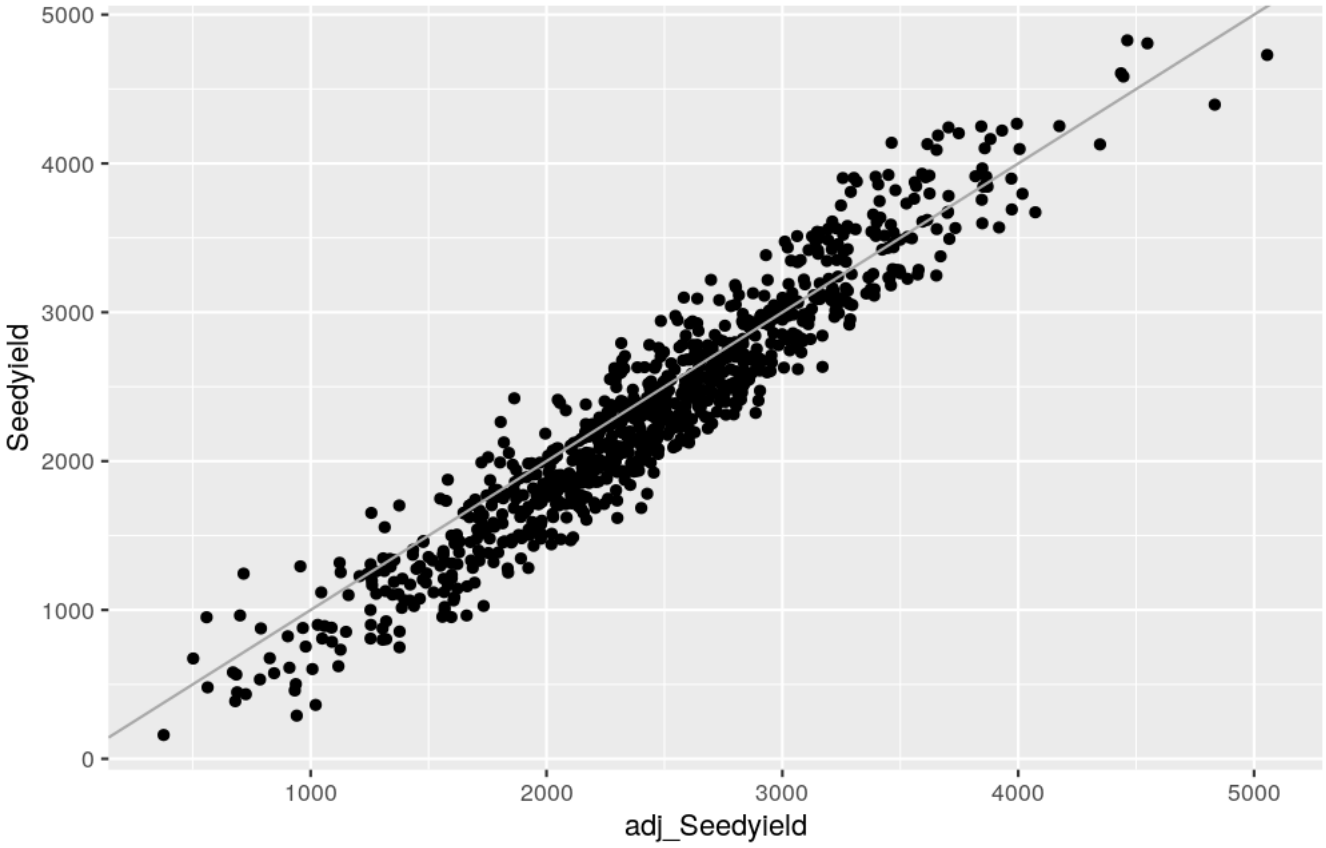
```
Glob_res <- remlf90(
  fixed  =  Seedyield ~ 1 + Year,
  random = ~  R8, # If you need random effects
  generic = list(genetic = list(inc.H,HX),
                 sp18 = list(inc.sp18,
                             breedR:::get_structure(sp18)),
                 sp19 = list(inc.sp19,
                             breedR:::get_structure(sp19))
  ),
  data = phenoSoy,
  method = 'em'
)

psp18V <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp18),
    z = as.vector(m18 %*% ranef(Glob_res)$sp18)
  )
)
psp19V <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp19),
    z = as.vector(m19 %*% ranef(Glob_res)$sp19)
  )
)

summary(Glob_res)
```

```
## Formula: Seedyield ~ 0 + Intercept + Year + R8
##    Data: phenoSoy
##    AIC     BIC logLik
##  13460 unknown  -6725
##
## Parameters of special components:
##
##
## Variance components:
##                 Estimated variances
## R8                            65770
## generic_genetic              162000
## sp18                          49310
## sp19                          37740
## Residual                     238800
##
## Fixed effects:
##             value   s.e.
## Intercept 2391.47 131.27
## Year.2018    0.00   0.00
## Year.2019 -233.13 165.47
```

```
compare.plots(list("2018V" = psp18V,
                   "2019V" = psp19V))

Glob_res$var["generic_genetic",1]/sum(Glob_res$var[,1]) #heritability
```

```
## [1] 0.2926195
```

```
phenoSoy$adj_Seedyield <- phenoSoy$Seedyield -
  as.vector(model.matrix(Glob_res)$R8 %*% ranef(Glob_res)$R8)

phenoSoy$adj_Seedyield[phenoSoy$Year == "2018"] =
  phenoSoy$adj_Seedyield[phenoSoy$Year == "2018"] -
  Glob_res$fixed$Year[[1]]["2018","value"]

phenoSoy$adj_Seedyield[phenoSoy$Year == "2019"] =
  phenoSoy$adj_Seedyield[phenoSoy$Year == "2019"] -
  Glob_res$fixed$Year[[1]]["2019","value"]

phenoSoy$adj_Seedyield[phenoSoy$Year == "2018"] =
  phenoSoy$adj_Seedyield[phenoSoy$Year == "2018"] -
  as.vector(m18 %*% ranef(Glob_res)$sp18)

phenoSoy$adj_Seedyield[phenoSoy$Year == "2019"] =
  phenoSoy$adj_Seedyield[phenoSoy$Year == "2019"] -
  as.vector(m19 %*% ranef(Glob_res)$sp19)

ggplot(phenoSoy, aes(adj_Seedyield, Seedyield)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, col = 'darkgray')
```




```
Glob_res_adj <- remlf90(
  fixed  =  adj_Seedyield ~ 1 + Year,
  random = ~  R8, # If you need random effects
  generic = list(genetic = list(inc.H,HX),
                 sp18 = list(inc.sp18,
                             breedR:::get_structure(sp18)),
                 sp19 = list(inc.sp19,
                             breedR:::get_structure(sp19))
  ),
  data = phenoSoy,
  method = 'em'
)


summary(Glob_res_adj)
```

```
## Formula: adj_Seedyield ~ 0 + Intercept + Year + R8
##    Data: phenoSoy
##    AIC     BIC logLik
##  13345 unknown  -6668
##
## Parameters of special components:
##
##
## Variance components:
##                 Estimated variances
```

```
## R8                             61.06
## generic_genetic           158900.00
## sp18                         226.60
## sp19                         242.30
## Residual                  227200.00
##
## Fixed effects:
##                 value   s.e.
## Intercept 2391.245328 32.685
## Year.2018    0.000000  0.000
## Year.2019   -0.089497 34.815
```

```
Glob_res_adj$var["generic_genetic",1]/sum(Glob_res_adj$var[,1]) #heritability
```

```
## [1] 0.4109873
```

The graph shows the phenotypic value depending on the adjusted phenotypic value.

Proteincontent

The model :

$Y_{ijk} = \mu + s_{kl} + g_j + a_i + m_{ijk} + e_{ijk}$

- Y_{ijk} = phenotype of the j^{th} genotype in the i^{th} year of the k^{th} spatial location (row and column)
- μ : overall mean
- s_k : the effect of the k^{th} spatial location
- g_j : the genetic effect of the j^{th} genotype
- m_{ijk} : the maturity of the j^{th} genotype in the i^{th} year of the k^{th} spatial location (row and column)
- e_{ijk} : the residual

The year is defined as fixed effect, the rest are random effects.

```
Glob_res <- remlf90(
  fixed  =  Proteincontent ~ 1 + Year,
  random = ~  R8, # If you need random effects
  generic = list(genetic = list(inc.H,HX),
                 sp18 = list(inc.sp18,
                             breedR:::get_structure(sp18)),
                 sp19 = list(inc.sp19,
                             breedR:::get_structure(sp19))
  ),
  data = phenoSoy,
  method = 'em'
)

psp18V <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp18),
    z = as.vector(m18 %*% ranef(Glob_res)$sp18)
  )
)
psp19V <- breedR:::spatial.plot(
  dat = data.frame(
    coordinates(sp19),
    z = as.vector(m19 %*% ranef(Glob_res)$sp19)
```

```
  )
)

summary(Glob_res)
```

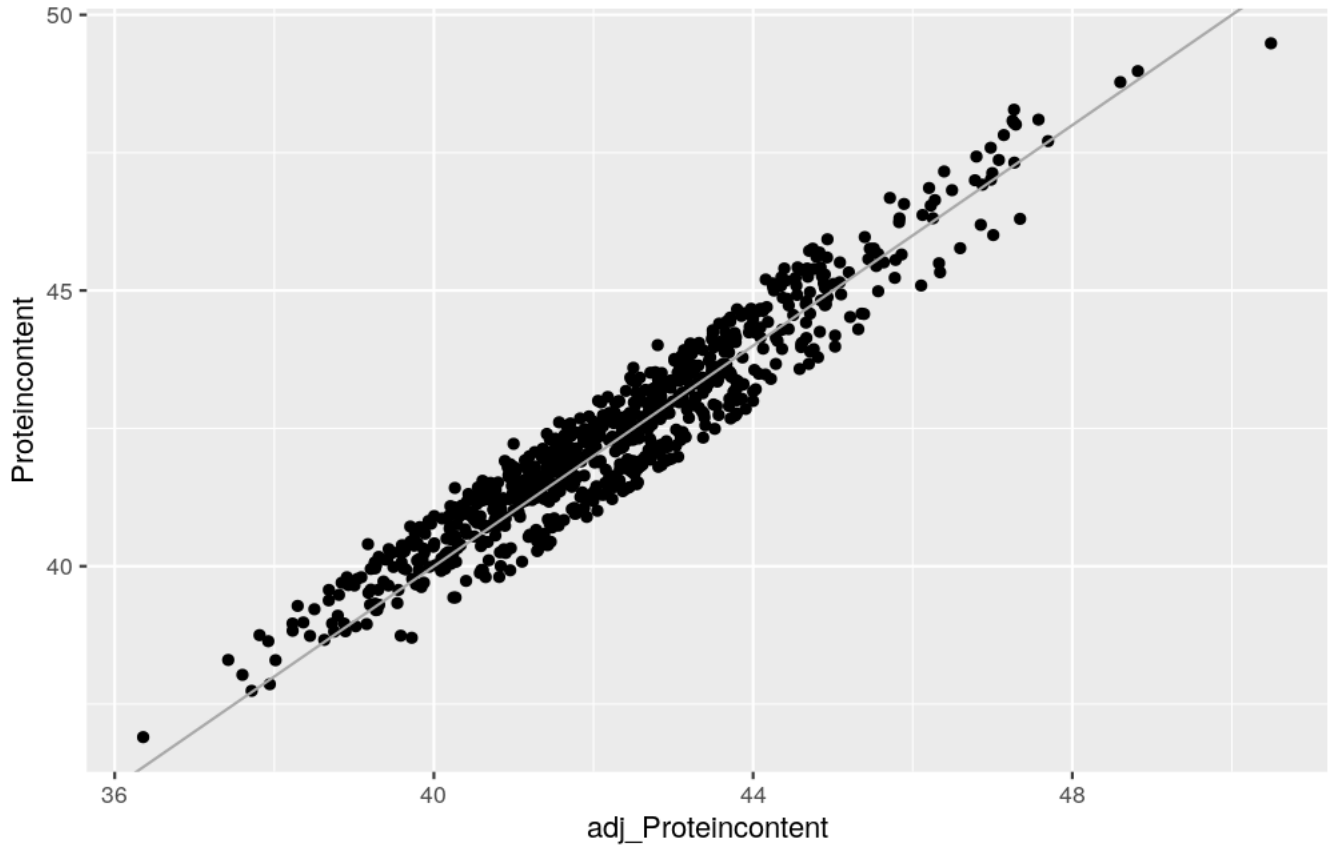
```
## Formula: Proteincontent ~ 0 + Intercept + Year + R8
##    Data: phenoSoy
##   AIC     BIC logLik
##  2831 unknown  -1411
##
## Parameters of special components:
##
##
## Variance components:
##                  Estimated variances
## R8                           0.277700
## generic_genetic              2.733000
## sp18                         0.007393
## sp19                         0.086720
## Residual                     0.774900
##
## Fixed effects:
##              value   s.e.
## Intercept 42.38211 0.1628
## Year.2018  0.00000 0.0000
## Year.2019  0.43911 0.2147
```

```
compare.plots(list("2018V" = psp18V,
                   "2019V" = psp19V))

Glob_res$var["generic_genetic",1]/sum(Glob_res$var[,1]) #heritability
```

```
## [1] 0.7044335
```

```
phenoSoy$adj_Proteincontent <- phenoSoy$Proteincontent -
  as.vector(model.matrix(Glob_res)$R8 %*% ranef(Glob_res)$R8)

phenoSoy$adj_Proteincontent[phenoSoy$Year == "2018"] =
  phenoSoy$adj_Proteincontent[phenoSoy$Year == "2018"] -
  Glob_res$fixed$Year[[1]]["2018","value"]

phenoSoy$adj_Proteincontent[phenoSoy$Year == "2019"] =
  phenoSoy$adj_Proteincontent[phenoSoy$Year == "2019"] -
  Glob_res$fixed$Year[[1]]["2019","value"]

phenoSoy$adj_Proteincontent[phenoSoy$Year == "2018"] <-
  phenoSoy$adj_Proteincontent[phenoSoy$Year == "2018"] -
  as.vector(m18 %*% ranef(Glob_res)$sp18)

phenoSoy$adj_Proteincontent[phenoSoy$Year == "2019"] <-
  phenoSoy$adj_Proteincontent[phenoSoy$Year == "2019"] -
  as.vector(m19 %*% ranef(Glob_res)$sp19)

ggplot(phenoSoy, aes(adj_Proteincontent, Proteincontent)) +
  geom_point() +
```

```
  geom_abline(intercept = 0, slope = 1, col = 'darkgray')
```




```
Glob_res_adj <- remlf90(
  fixed  =  adj_Proteincontent ~ 1 + Year,
  random = ~  R8, # If you need random effects
  generic = list(genetic = list(inc.H,HX),
                 sp18 = list(inc.sp18,
                             breedR:::get_structure(sp18)),
                 sp19 = list(inc.sp19,
                             breedR:::get_structure(sp19))
  ),
  data = phenoSoy,
  method = 'em'
)


summary(Glob_res_adj)
```

```
## Formula: adj_Proteincontent ~ 0 + Intercept + Year + R8
##    Data: phenoSoy
##   AIC     BIC logLik
##  2743 unknown  -1367
##
## Parameters of special components:
##
```

```
##
## Variance components:
##                  Estimated variances
## R8                          0.0003249
## generic_genetic             2.7300000
## sp18                        0.0025690
## sp19                        0.0015620
## Residual                    0.7363000
##
## Fixed effects:
##                  value   s.e.
## Intercept 42.38462454 0.1021
## Year.2018  0.00000000 0.0000
## Year.2019 -0.00021481 0.0709
```

```
Glob_res_adj$var["generic_genetic",1]/sum(Glob_res_adj$var[,1]) #heritability
```

```
## [1] 0.7865722
```

Genotypic mean :

```
Adj_PhenoSoy = phenoSoy %>% dplyr::select(EUCLEGID,
    adj_Proteincontent, adj_Seedyield) %>% group_by(EUCLEGID) %>%
    dplyr::summarise_each(list(~mean(., na.rm = TRUE))) %>%
    as.data.frame(strings.As.factor = F)

summary(Adj_PhenoSoy)
```

```
##    EUCLEGID         adj_Proteincontent adj_Seedyield
##  Length:360         Min.   :37.72      Min.   : 580.9
##  Class :character   1st Qu.:41.15      1st Qu.:2031.1
##  Mode  :character   Median :42.17      Median :2358.8
##                     Mean   :42.36      Mean   :2389.0
##                     3rd Qu.:43.44      3rd Qu.:2775.4
##                     Max.   :49.09      Max.   :4503.9
##                     NA's   :3
```

Congrats ! You have performed your first genomic evaluation !!! There is a difference between the genomic evaluation and the genomic prediction, even some authors in the literature tend to merge the two.

Genomic evaluation consist in using the marker to estimate the genetic value of the individuals.

Genomic prediction means you don't know the phenotype (or genetic value) of the individuals and by using the marker you will estimate it.

It is what we will do after the GWAS part.

GWAS

The GWAS can be used in several situations. Firstly, when seeking to determine the genetic architecture of a trait of ecological or agronomic interest. It is also useful in the context of Marker Assisted Breeding and Genomic Selection. GWAS aims at determine the association between the phenotype and genotype of individuals in a population by the way of linear regression. Some parameters influence the GWAS results :

- LD : Non random association between alleles from different (linked) loci
- Population structure : Allele frequencies vary across sub-populations and cause long-range disequilibrium, i.e. between unlinked loci

Choice of the model

Several methods have been developed over the years to perform GWAS :

- GRAMMAR (Aulchenko et al., 2007)
- P3D (Zhang et al., 2010) & EMMAX (Kang et al., 2010)
- FaST-LMM (Lippert et al., 2011)
- GEMMA (Zhou & Stephens, 2012)
- MixABEL (Aulchenko, et al.)
- CMLM (Zhang et a., 2010)
- ECMLM (Li et al., 2014)
- MLMM (Segura et al., 2012)
- FarmCPU (Liu et al, 2016)
- ...

A very good illustration in the paper of Liu et al, 2016 shows the differences between the methods. Here is the figure and the legend of this one taken directly from the publication of Liu et al, 2016

We are going to test the MLMM with the genomic relationship matrix. As the genomic relationship matrix includes information on both population structure and relatedness, it is in general not useful to consider admixture information as fixed effects covariates (Astle and Balding 2009).

This method is a stepwise model regression with a forward inclusion and a backward elimination. This limits the space of the model to be explored and makes the method computationally efficient.

Protein content

First we are using the mlmm method from the package *mlmm*, I did some tests to define the number of steps to use.

I change the coding of the chromosome to allow easier graphic representation.

```
colnames(Map) = c("SNP", "Chr", "Pos")
Map$Chr2 = Map$Chr  # I keep the chromosome code in another column
# I only keep the chromosome or scaffold number.
Map$Chr[grep("Gm", Map$Chr)] = as.numeric(sub("Gm",
    "", Map$Chr[grep("Gm", Map$Chr)]))
Map$Chr[grep("scaffold_", Map$Chr)] = as.numeric(sub("scaffold_",
    "", Map$Chr[grep("scaffold_", Map$Chr)]))
Map$Chr = as.numeric(Map$Chr)  # Transforme the variable from character to numeric
# I only keep the markers present in the genotyping
# matrix.
Map = Map[Map$SNP %in% colnames(M), ]

Map$id = 1:nrow(Map)

# I remove individuals with missing data for the
# trait in question.
data_gwas = na.omit(Adj_PhenoSoy[match(rownames(GV),
    Adj_PhenoSoy$EUCLEGID), c("EUCLEGID", "adj_Proteincontent")])

# I store and order the molecular matching matrix
# calculated using the VanRaden method according to
# the matrix with the raw genotyping.

K = GV[match(data_gwas$EUCLEGID, rownames(GV)), match(data_gwas$EUCLEGID,
```

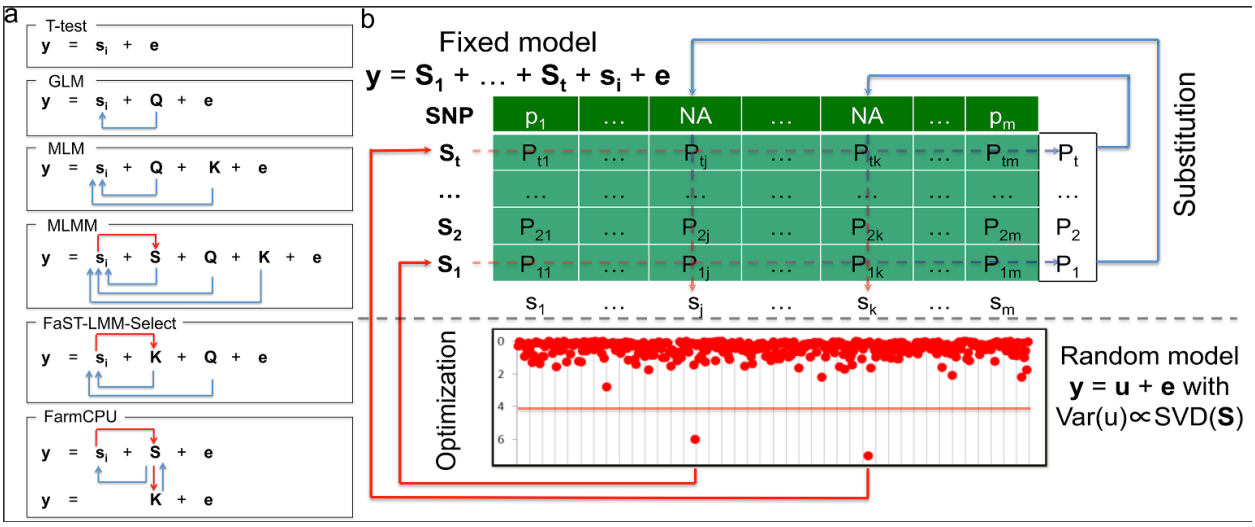


Figure 2: Comparison of the different proposed method for GWAS in the left panel (a). These methods start with a naïve model (e.g. t-test) that tests marker effect, one at a time, i.e. ith marker (si), on the phenotype (y) with a residual effect (e). Next, GLM controls false positives by fitting population structure (Q) as covariates to adjust the test on genetic markers indicated by the blue arrows. MLM fits both Q and kinship (K) as covariates. However, both Q and K remain constant for testing all the markers. Neither Q nor K receives adjustment from association tests on markers. MLMM add pseudo QTNs as additional covariates (S). These pseudo QTNs are estimated through a stepwise regression procedure. Consequently, these pseudo QTNs receive adjustment from association tests on markers as indicated by the red arrow. However, both Q and K remain constant for testing all the markers. Although similar to MLM, FaST-LMM-Select controls false positives by fitting Q and K as covariates; the K of FaST-LMM-Select is incorporated with association tests on markers as indicated by the red arrow. However, Q remains constant. FarmCPU completely removes the confounding between the testing marker and both K and Q by combining MLMM and FaST-LMM-Select, but allowing a fixed effect model and a random effect model to perform separately. The fixed effect model contains the testing marker and pseudo QTNs to control false positives. The pseudo QTNs are selected from associated markers and evaluated by the random effect model, with K defined by the pseudo QTNs. The fixed effect model and random effect model are used iteratively until a stage of convergence is reached, that is, when no new pseudo QTNs are added. The right panel (b) displays the fixed effect model above the dashed line and the random effect models below the dashed line. The t pseudo QTNs (S1 to St) are fitted as covariates to test markers one at a time, e.g., ith marker (si) in the fixed model. As the pseudo QTNs are fitted as covariates for each marker, Not Available (NA) is assigned as the test statistic for all markers that are also pseudo QTNs—as the genetic marker is completely co-linear to the pseudo QTN marker. However, each pseudo QTN has a test statistic corresponding to every marker, creating a matrix (lightly shaded) with elements of Pij, i = 1 to t and j = 1 to m. The most significant P value of each pseudo QTN (the vector on the right of shaded area) is used as the substitution for the NA of the corresponding marker. The pseudo QTNs are optimized by using the SUPER method in the random model to incorporate both test statistics from the fixed effect model and genetic map information in the genotype data. The random effects are the individuals' genetic effects (u) with variance and covariance matrix, Var(u), defined by the Singular Value Decomposition (SVD) on the pseudo QTNs by using the FaST-LMM algorithm. The updated set of pseudo QTNs go back into the fixed model. The process continuously repeats until no more pseudo QTNs are added.

```
    colnames(GV))]
dim(data_gwas)
```

```
## [1] 354   2
```

```
dim(K)
```

```
## [1] 354 354
```

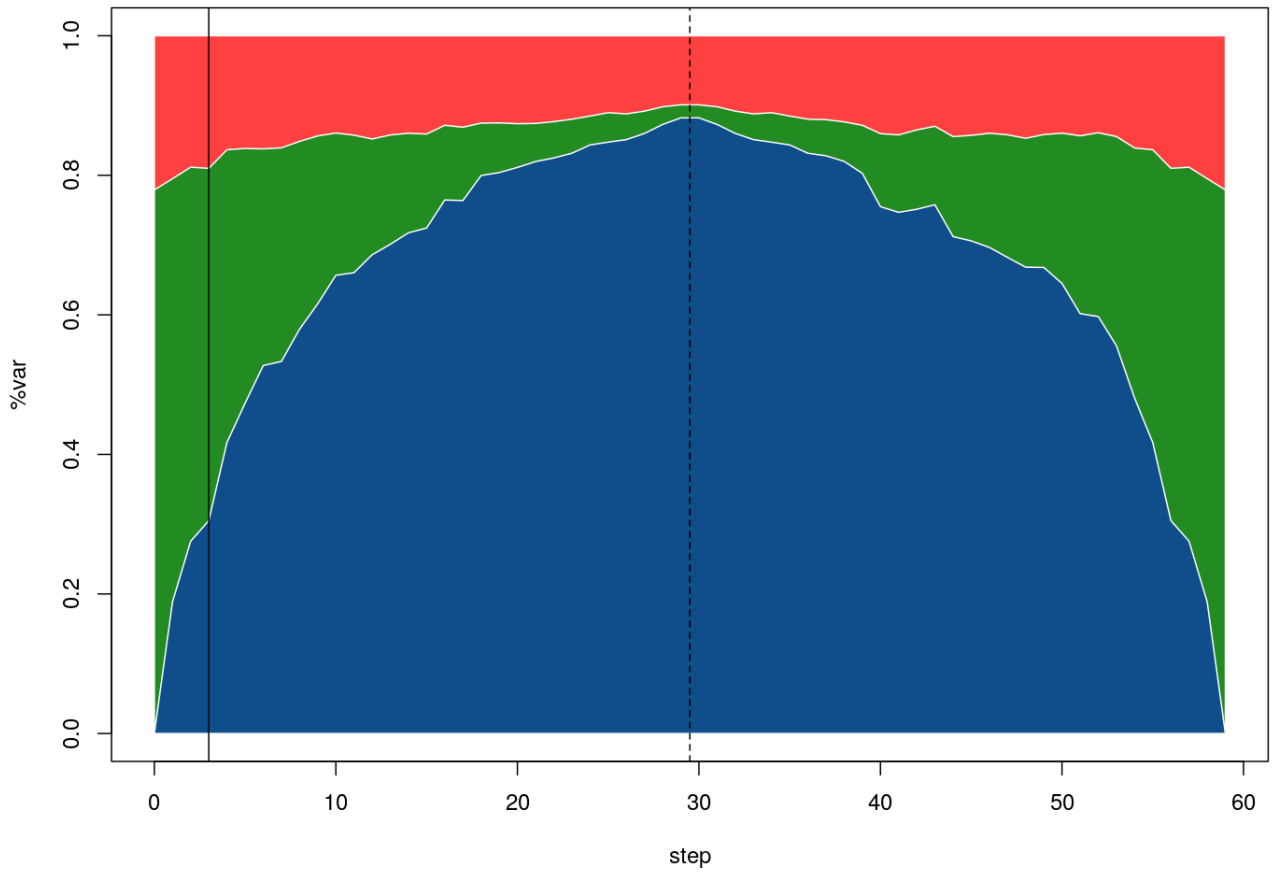
```
# The package requires a genotyping coding in 0,1,2
# so I change the raw data.
C = M[match(data_gwas$EUCLEGID, rownames(M)), ] + 1

# Launch of QTL detection
mygwas_proteincontent <- mlmm(Y = data_gwas$adj_Proteincontent,
    X = C, K = K, nbchunks = 2, maxsteps = 30)
```

```
## null model done! pseudo-h= 0.779
## step 1 done! pseudo-h= 0.748
## step  2  done! pseudo-h= 0.74
## step  3  done! pseudo-h= 0.727
## step  4  done! pseudo-h= 0.72
## step  5  done! pseudo-h= 0.693
## step  6  done! pseudo-h= 0.657
## step  7  done! pseudo-h= 0.656
## step  8  done! pseudo-h= 0.64
## step  9  done! pseudo-h= 0.626
## step  10  done! pseudo-h= 0.594
## step  11  done! pseudo-h= 0.58
## step  12  done! pseudo-h= 0.528
## step  13  done! pseudo-h= 0.524
## step  14  done! pseudo-h= 0.505
## step  15  done! pseudo-h= 0.489
## step  16  done! pseudo-h= 0.455
## step  17  done! pseudo-h= 0.445
## step  18  done! pseudo-h= 0.375
## step  19  done! pseudo-h= 0.363
## step  20  done! pseudo-h= 0.332
## step  21  done! pseudo-h= 0.303
## step  22  done! pseudo-h= 0.297
## step  23  done! pseudo-h= 0.29
## step  24  done! pseudo-h= 0.265
## step  25  done! pseudo-h= 0.277
## step  26  done! pseudo-h= 0.248
## step  27  done! pseudo-h= 0.228
## step  28  done! pseudo-h= 0.199
## step  29  done! pseudo-h= 0.159
## backward analysis
## creating output
```

Graph showing the variance explained by the SNPs at each stage. In red the residual variance, in green the genetic variance and in blue the variance explained by the QTLs detected. We can see here that the explained variance is increasing. To choose the best step, you can take the one that gives the highest explained variance or use criteria such as the BIC and/or Bonferroni. The vertical black line represents the step chosen by the Bonferonni or BIC threshold. Using these two criteria as thresholds ensures that there are no false positives, but they are very conservative and lead to the loss of detected QTLs. There are several schools, those that

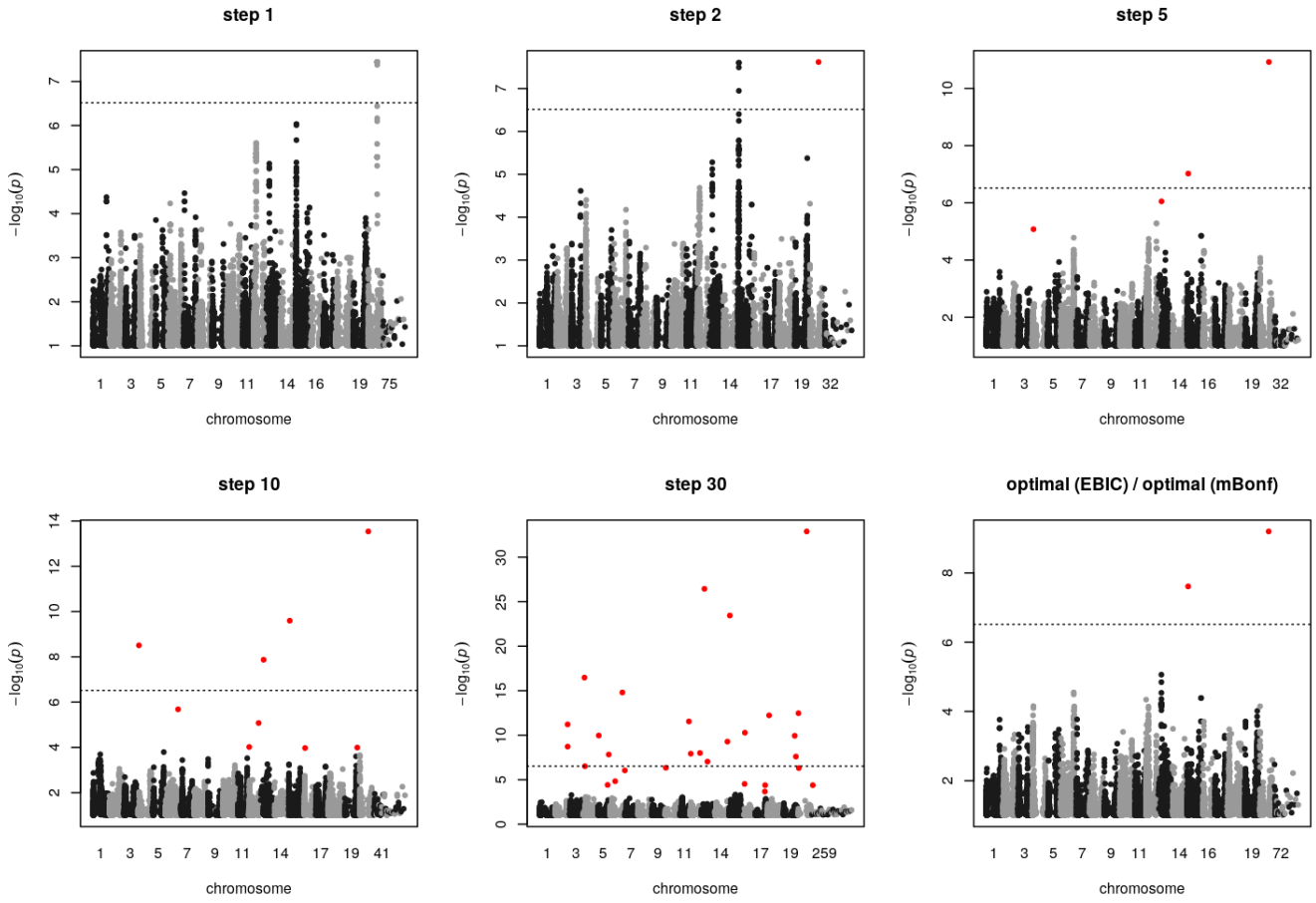
use the thresholds even if nothing is detected and those that present both results.

```
plot_step_RSS(mygwas_proteincontent)  # % variance plot
abline(v = which.min(mygwas_proteincontent$step_table$extBIC),
    col = "Black", cex = 3)
```



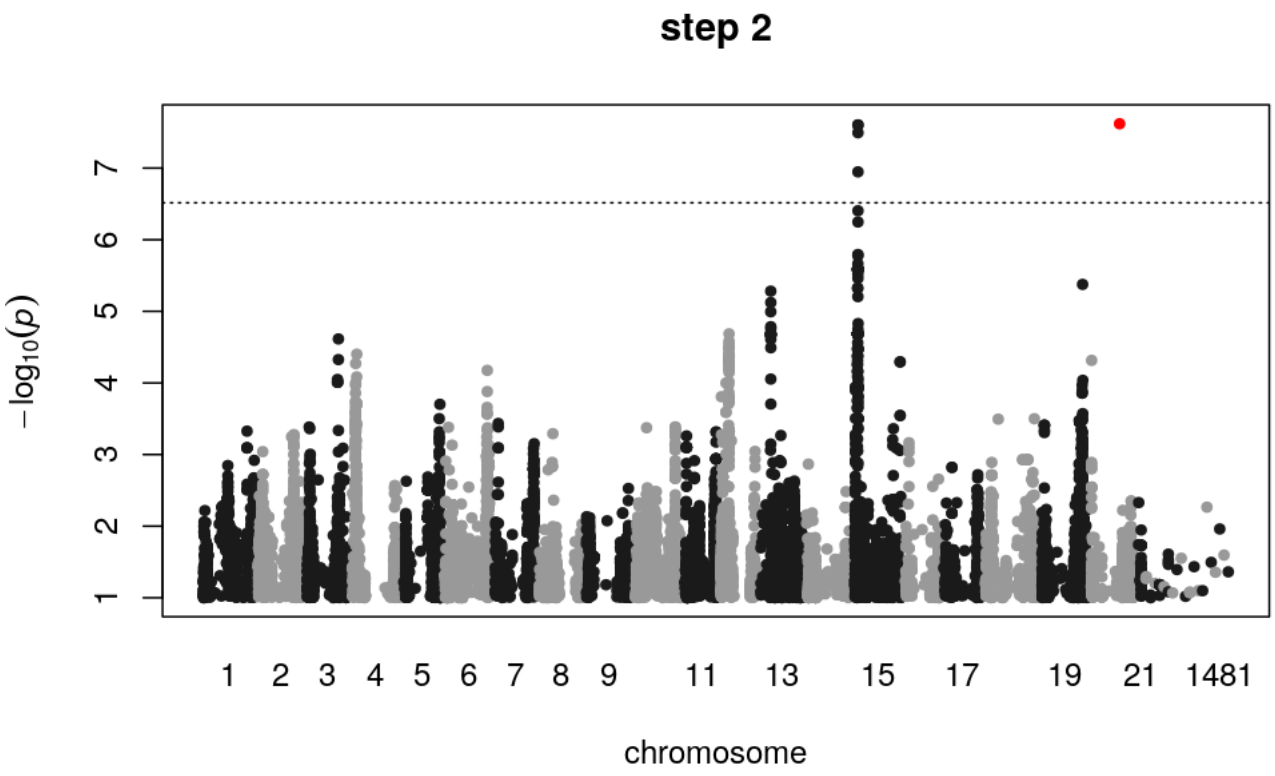
Representation of the QTLs detected with the first step of the MLMM. all SNPs are represented on the chromosomes and on the scaffolds. the dotted horizontal line represents the Bonferroni thresholds.

```
plot_fwd_GWAS(mygwas_proteincontent, 1, Map[, 1:3],
    1, main = "step 1")  # 1st mlmm step plot
```



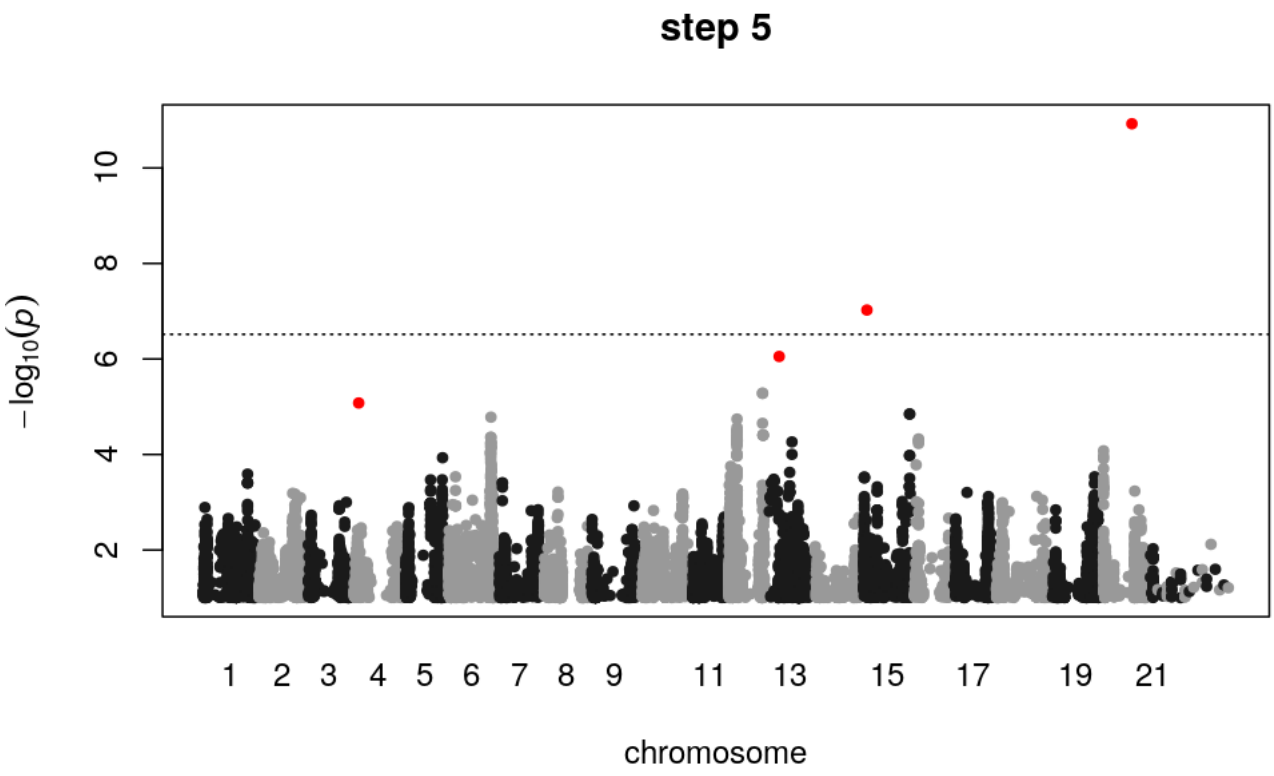
Representation of the QTLs detected with the second step of the MLMM. Only SNPs with a p-value greater than 0.1 are displayed in the graph to simplify it. SNPs detected by the method are shown in red. The dotted horizontal line represents the Bonferroni thresholds.

```
plot_fwd_GWAS(mygwas_proteincontent, 2, Map[, 1:3],
    0.1, main = "step 2")  # 2nd mlmm step plot
```



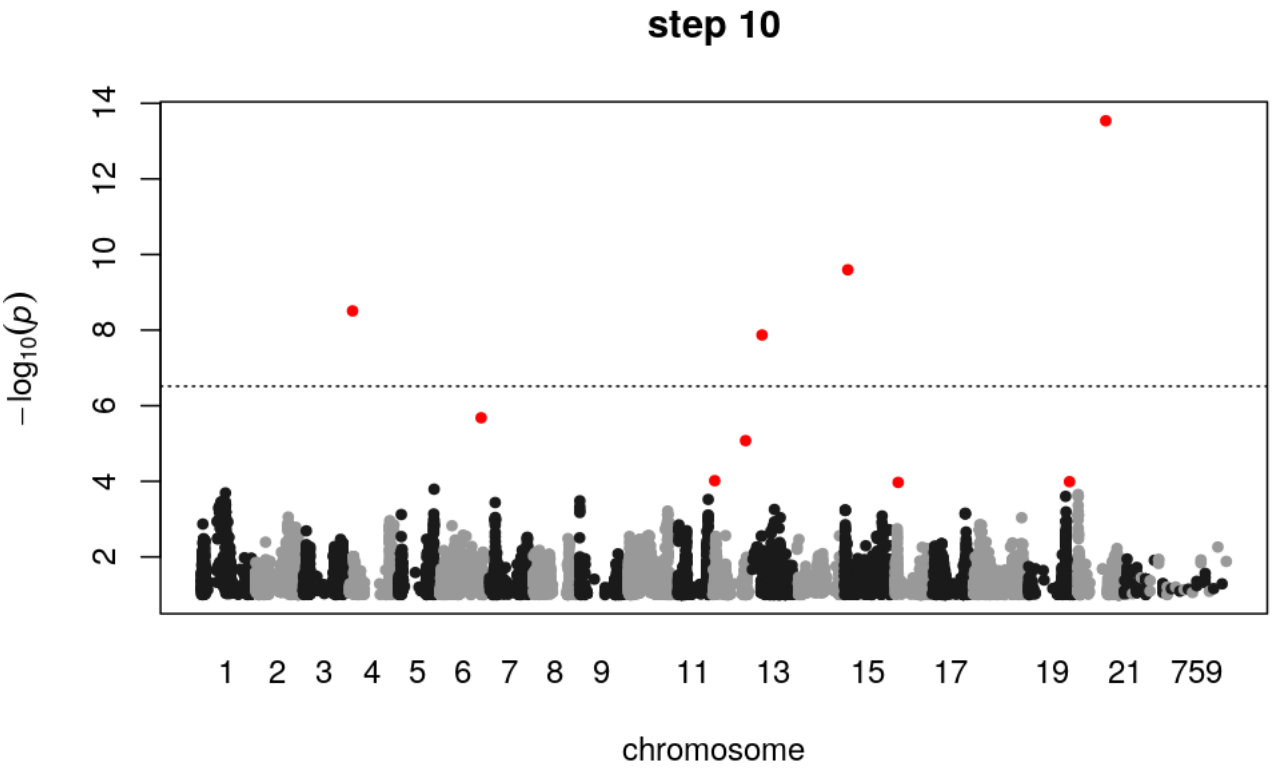
Representation of the QTLs detected with the fifth step of the MLMM. Only SNPs with a p-value greater than 0.1 are displayed in the graph to simplify it. SNPs detected by the method are shown in red. The dotted horizontal line represents the Bonferroni thresholds.

```
plot_fwd_GWAS(mygwas_proteincontent, 5, Map[, 1:3],
    0.1, main = "step 5")  # 3rd mlmm step plot
```



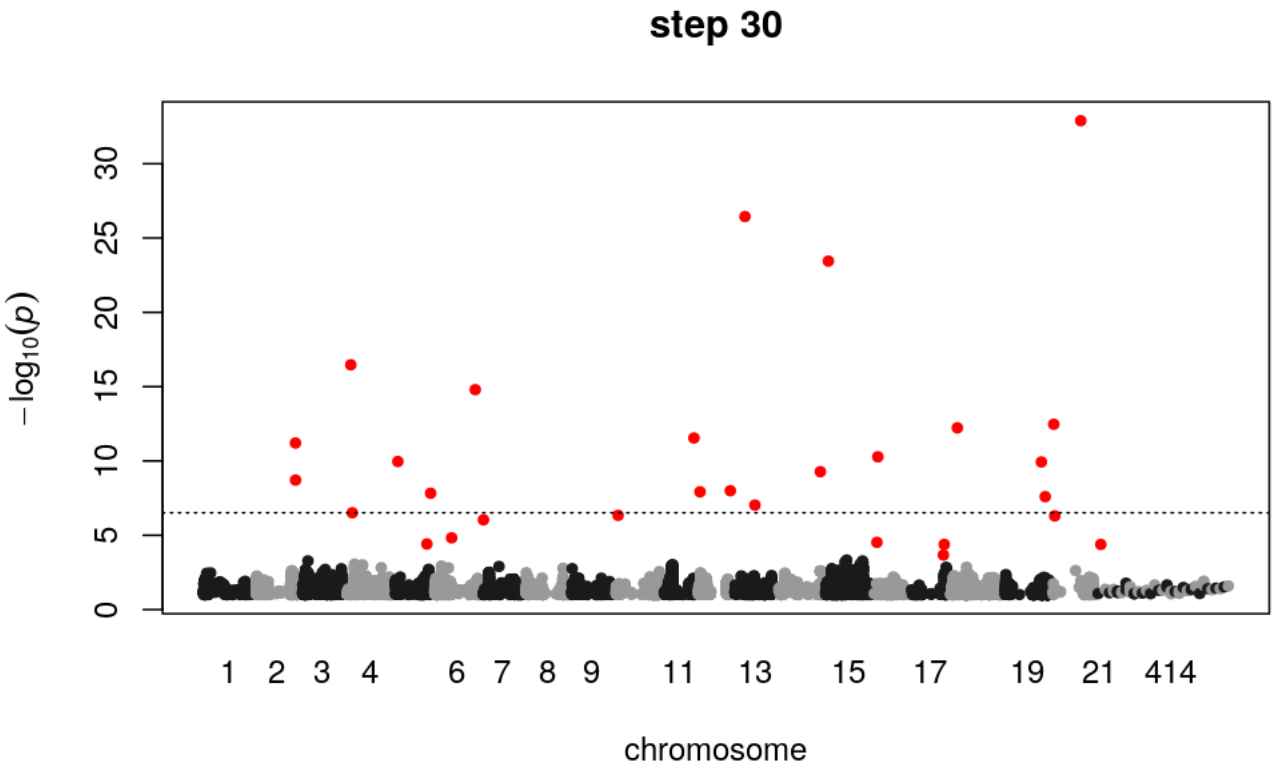
Representation of the QTLs detected with the tenth step of the MLMM. Only SNPs with a p-value greater than 0.1 are displayed in the graph to simplify it. SNPs detected by the method are shown in red. The dotted horizontal line represents the Bonferroni thresholds.

```
plot_fwd_GWAS(mygwas_proteincontent, 10, Map[, 1:3],
    0.1, main = "step 10")  # 3rd mlmm step plot
```



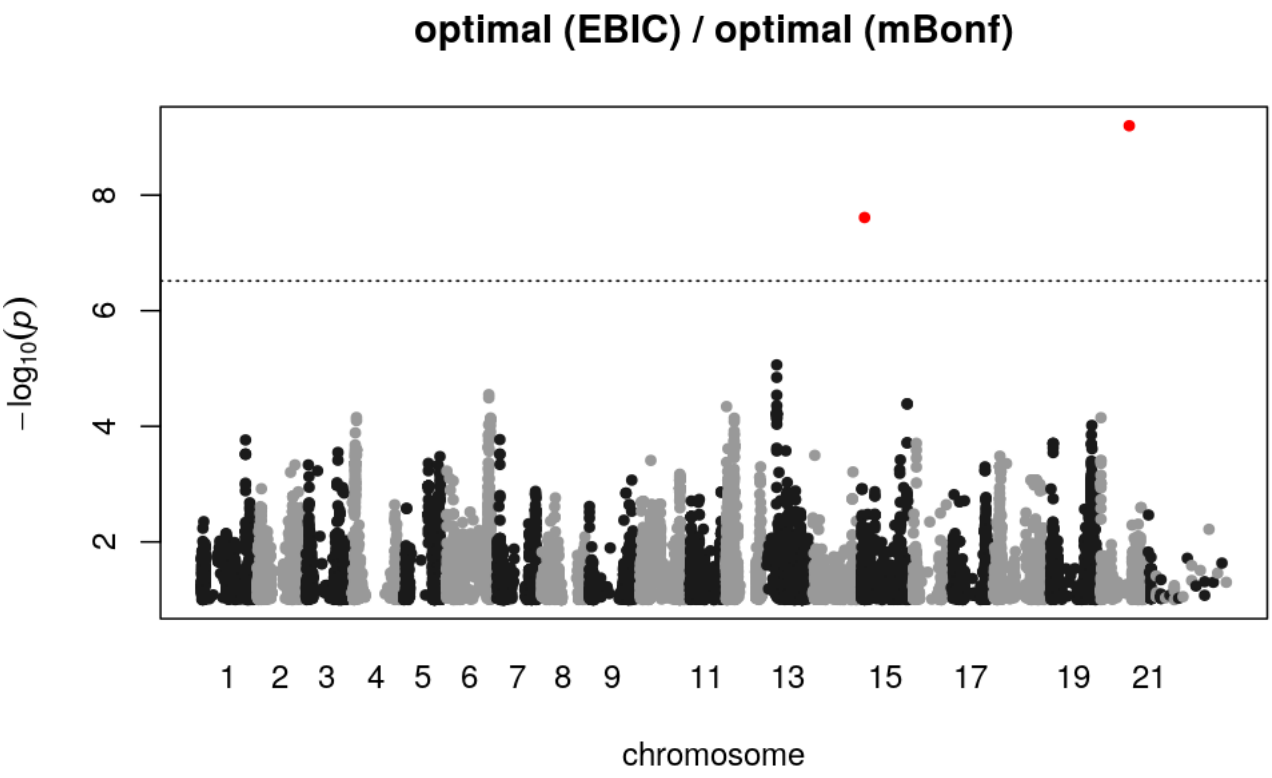
Representation of the QTLs detected with the thirtieth step of the MLMM. Only SNPs with a p-value greater than 0.1 are displayed in the graph to simplify it. SNPs detected by the method are shown in red. The dotted horizontal line represents the Bonferroni thresholds.

```
plot_fwd_GWAS(mygwas_proteincontent, 30, Map[, 1:3],
    0.1, main = "step 30")  # 3rd mlmm step plot
```



Representation of the QTLs detected in the third stage of the MLMM, this stage is chosen based on the BIC criterion and the Bonferroni threshold.

```
plot_opt_GWAS(mygwas_proteincontent, "extBIC", Map[,
    1:3], 0.1, main = "optimal (EBIC) / optimal (mBonf)")
```




```
# optimal step according to eBIC plot
```

The more steps you add to the mlmm, the more QTL's come out of the handle. It is up to the analyst to choose the step that seems the most relevant and to use the threshold of his choice, or to set up methods for validating the QTLs.

Gapit3 can also perform mlmm . GAPIT gives a relatively crude result, it is up to us to decide the threshold for the QTLs detected. I did not find the information for the step at which GAPIT stopped.

```
myGAPIT_MLMM_proteincontent <- GAPIT(Y = data_gwas[,
    c(1, 2)], GD = cbind(data.frame(rownames(C), stringsAsFactors = F),
    C), GM = Map[, 1:3], model = "mlmm", file.output = F)
```

```
## [1] "---------------------- Welcome to GAPIT -----------------------------"
## [1] "All packages are loaded already !  GAPIT.Version is 2018.08.18, GAPIT 3.0"
## [1] "mlmm"
## [1] "--------------------Processing traits----------------------------------"
## [1] "Phenotype provided!"
## [1] "The 1 model in all."
## [1] "mlmm"
## [1] "GAPIT.DP in process..."
## [1] "GAPIT will filter marker with MAF setting !!"
## [1] "The markers will be filtered by SNP.MAF: 0"
## maf_index
##   TRUE
## 163935
## [1] "Calculating kinship..."
## [1] "Number of individuals and SNPs are  354  and  163935"
```

```
## [1] "Calculating kinship with VanRaden method..."
## [1] "substracting P..."
## [1] "Getting X'X..."
## [1] "Adjusting..."
## [1] "Calculating kinship with VanRaden method: done"
## [1] "kinship calculated"
## [1] "Adding IDs to kinship..."
## [1] "Writing kinship to file..."
## [1] "Kinship save as file"
## [1] "Kinship created!"
## NULL
## [1] "GAPIT.DP accomplished successfully for multiple traits. Results are saved"
## [1] "Processing trait: adj_Proteincontent"
## [1] "--------------------Phenotype and Genotype ----------------------------------"
## [1] "VanRaden"
## [1] "There are  1  traits in phenotype data."
## [1] "There are  354  individuals in phenotype data."
## [1] "There are  163935  markers in genotype data."
## [1] "Phenotype and Genotype are test OK !!"
## [1] "--------------------GAPIT Logical ----------------------------------"
## [1] "GAPIT.IC in process..."
## [1] "There is 0 Covarinces."
## [1] "There are 354 common individuals in genotype , phenotype and CV files."
## [1] "The dimension of total CV is "
## [1] 354   2
## [1] "GAPIT.IC accomplished successfully for multiple traits. Results are saved"
## [1] "GAPIT.SS in process..."
## [1] "------------Examining data (QC)----------------------------------------"
## [1] "QC is in process..."
## [1] "Removing duplicates..."
## [1] "Removing NaN..."
## [1] "Remove duplicates for GT..."
## [1] "Remove duplicates for KI..."
## [1] "Remove duplicates for Z (column wise)..."
## [1] "Maching Z with Kinship colwise..."
## [1] "Maching Z without origin..."
## [1] "Maching GT and CV..."
## [1] "QC final process..."
## [1] "GAPIT.QC accomplished successfully!"
## [1] "Try to group from and to were set to 1"
## [1] "------------Examining data (QC) done-----------------------------------"
## [1] "--------------------Sandwich burger and dressing-------------------------"
## [1] "------------------------Iteration in process--------------------------"
## [1] "Total iterations: 1"
## [1] "Compressing and Genome screening..."
## [1] "-------Mixed model with Kinship-----------------------------"
## [1] "Genotype file: 1, SNP: 1000 "
## [1] "Genotype file: 1, SNP: 2000 "
## [1] "Genotype file: 1, SNP: 3000 "
## [1] "Genotype file: 1, SNP: 4000 "
## [1] "Genotype file: 1, SNP: 5000 "
## [1] "Genotype file: 1, SNP: 6000 "
## [1] "Genotype file: 1, SNP: 7000 "
## [1] "Genotype file: 1, SNP: 8000 "
```

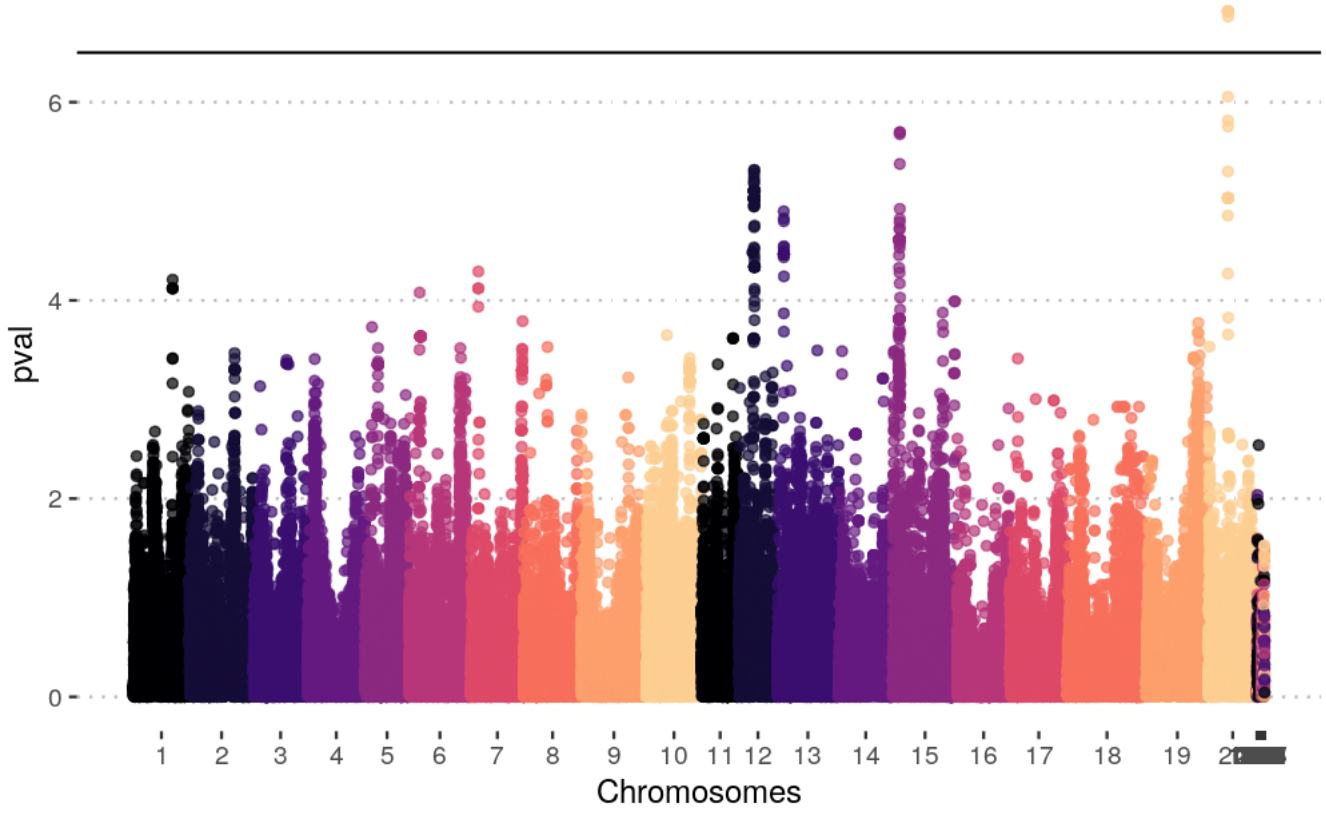
```
## [1] "Genotype file: 1, SNP: 9000 "
## [1] "Genotype file: 1, SNP: 10000 "
## [1] "Genotype file: 1, SNP: 11000 "
## [1] "Genotype file: 1, SNP: 12000 "
## [1] "Genotype file: 1, SNP: 13000 "
## [1] "Genotype file: 1, SNP: 14000 "
## [1] "Genotype file: 1, SNP: 15000 "
## [1] "Genotype file: 1, SNP: 16000 "
## [1] "Genotype file: 1, SNP: 17000 "
## [1] "Genotype file: 1, SNP: 18000 "
## [1] "Genotype file: 1, SNP: 19000 "
## [1] "Genotype file: 1, SNP: 20000 "
## [1] "Genotype file: 1, SNP: 21000 "
## [1] "Genotype file: 1, SNP: 22000 "
## [1] "Genotype file: 1, SNP: 23000 "
## [1] "Genotype file: 1, SNP: 24000 "
## [1] "Genotype file: 1, SNP: 25000 "
## [1] "Genotype file: 1, SNP: 26000 "
## [1] "Genotype file: 1, SNP: 27000 "
## [1] "Genotype file: 1, SNP: 28000 "
## [1] "Genotype file: 1, SNP: 29000 "
## [1] "Genotype file: 1, SNP: 30000 "
## [1] "Genotype file: 1, SNP: 31000 "
## [1] "Genotype file: 1, SNP: 32000 "
## [1] "Genotype file: 1, SNP: 33000 "
## [1] "Genotype file: 1, SNP: 34000 "
## [1] "Genotype file: 1, SNP: 35000 "
## [1] "Genotype file: 1, SNP: 36000 "
## [1] "Genotype file: 1, SNP: 37000 "
## [1] "Genotype file: 1, SNP: 38000 "
## [1] "Genotype file: 1, SNP: 39000 "
## [1] "Genotype file: 1, SNP: 40000 "
## [1] "Genotype file: 1, SNP: 41000 "
## [1] "Genotype file: 1, SNP: 42000 "
## [1] "Genotype file: 1, SNP: 43000 "
## [1] "Genotype file: 1, SNP: 44000 "
## [1] "Genotype file: 1, SNP: 45000 "
## [1] "Genotype file: 1, SNP: 46000 "
## [1] "Genotype file: 1, SNP: 47000 "
## [1] "Genotype file: 1, SNP: 48000 "
## [1] "Genotype file: 1, SNP: 49000 "
## [1] "Genotype file: 1, SNP: 50000 "
## [1] "Genotype file: 1, SNP: 51000 "
## [1] "Genotype file: 1, SNP: 52000 "
## [1] "Genotype file: 1, SNP: 53000 "
## [1] "Genotype file: 1, SNP: 54000 "
## [1] "Genotype file: 1, SNP: 55000 "
## [1] "Genotype file: 1, SNP: 56000 "
## [1] "Genotype file: 1, SNP: 57000 "
## [1] "Genotype file: 1, SNP: 58000 "
## [1] "Genotype file: 1, SNP: 59000 "
## [1] "Genotype file: 1, SNP: 60000 "
## [1] "Genotype file: 1, SNP: 61000 "
## [1] "Genotype file: 1, SNP: 62000 "
```

```
## [1] "Genotype file: 1, SNP: 63000 "
## [1] "Genotype file: 1, SNP: 64000 "
## [1] "Genotype file: 1, SNP: 65000 "
## [1] "Genotype file: 1, SNP: 66000 "
## [1] "Genotype file: 1, SNP: 67000 "
## [1] "Genotype file: 1, SNP: 68000 "
## [1] "Genotype file: 1, SNP: 69000 "
## [1] "Genotype file: 1, SNP: 70000 "
## [1] "Genotype file: 1, SNP: 71000 "
## [1] "Genotype file: 1, SNP: 72000 "
## [1] "Genotype file: 1, SNP: 73000 "
## [1] "Genotype file: 1, SNP: 74000 "
## [1] "Genotype file: 1, SNP: 75000 "
## [1] "Genotype file: 1, SNP: 76000 "
## [1] "Genotype file: 1, SNP: 77000 "
## [1] "Genotype file: 1, SNP: 78000 "
## [1] "Genotype file: 1, SNP: 79000 "
## [1] "Genotype file: 1, SNP: 80000 "
## [1] "Genotype file: 1, SNP: 81000 "
## [1] "Genotype file: 1, SNP: 82000 "
## [1] "Genotype file: 1, SNP: 83000 "
## [1] "Genotype file: 1, SNP: 84000 "
## [1] "Genotype file: 1, SNP: 85000 "
## [1] "Genotype file: 1, SNP: 86000 "
## [1] "Genotype file: 1, SNP: 87000 "
## [1] "Genotype file: 1, SNP: 88000 "
## [1] "Genotype file: 1, SNP: 89000 "
## [1] "Genotype file: 1, SNP: 90000 "
## [1] "Genotype file: 1, SNP: 91000 "
## [1] "Genotype file: 1, SNP: 92000 "
## [1] "Genotype file: 1, SNP: 93000 "
## [1] "Genotype file: 1, SNP: 94000 "
## [1] "Genotype file: 1, SNP: 95000 "
## [1] "Genotype file: 1, SNP: 96000 "
## [1] "Genotype file: 1, SNP: 97000 "
## [1] "Genotype file: 1, SNP: 98000 "
## [1] "Genotype file: 1, SNP: 99000 "
## [1] "Genotype file: 1, SNP: 100000 "
## [1] "Genotype file: 1, SNP: 101000 "
## [1] "Genotype file: 1, SNP: 102000 "
## [1] "Genotype file: 1, SNP: 103000 "
## [1] "Genotype file: 1, SNP: 104000 "
## [1] "Genotype file: 1, SNP: 105000 "
## [1] "Genotype file: 1, SNP: 106000 "
## [1] "Genotype file: 1, SNP: 107000 "
## [1] "Genotype file: 1, SNP: 108000 "
## [1] "Genotype file: 1, SNP: 109000 "
## [1] "Genotype file: 1, SNP: 110000 "
## [1] "Genotype file: 1, SNP: 111000 "
## [1] "Genotype file: 1, SNP: 112000 "
## [1] "Genotype file: 1, SNP: 113000 "
## [1] "Genotype file: 1, SNP: 114000 "
## [1] "Genotype file: 1, SNP: 115000 "
## [1] "Genotype file: 1, SNP: 116000 "
```

```
## [1] "Genotype file: 1, SNP: 117000 "
## [1] "Genotype file: 1, SNP: 118000 "
## [1] "Genotype file: 1, SNP: 119000 "
## [1] "Genotype file: 1, SNP: 120000 "
## [1] "Genotype file: 1, SNP: 121000 "
## [1] "Genotype file: 1, SNP: 122000 "
## [1] "Genotype file: 1, SNP: 123000 "
## [1] "Genotype file: 1, SNP: 124000 "
## [1] "Genotype file: 1, SNP: 125000 "
## [1] "Genotype file: 1, SNP: 126000 "
## [1] "Genotype file: 1, SNP: 127000 "
## [1] "Genotype file: 1, SNP: 128000 "
## [1] "Genotype file: 1, SNP: 129000 "
## [1] "Genotype file: 1, SNP: 130000 "
## [1] "Genotype file: 1, SNP: 131000 "
## [1] "Genotype file: 1, SNP: 132000 "
## [1] "Genotype file: 1, SNP: 133000 "
## [1] "Genotype file: 1, SNP: 134000 "
## [1] "Genotype file: 1, SNP: 135000 "
## [1] "Genotype file: 1, SNP: 136000 "
## [1] "Genotype file: 1, SNP: 137000 "
## [1] "Genotype file: 1, SNP: 138000 "
## [1] "Genotype file: 1, SNP: 139000 "
## [1] "Genotype file: 1, SNP: 140000 "
## [1] "Genotype file: 1, SNP: 141000 "
## [1] "Genotype file: 1, SNP: 142000 "
## [1] "Genotype file: 1, SNP: 143000 "
## [1] "Genotype file: 1, SNP: 144000 "
## [1] "Genotype file: 1, SNP: 145000 "
## [1] "Genotype file: 1, SNP: 146000 "
## [1] "Genotype file: 1, SNP: 147000 "
## [1] "Genotype file: 1, SNP: 148000 "
## [1] "Genotype file: 1, SNP: 149000 "
## [1] "Genotype file: 1, SNP: 150000 "
## [1] "Genotype file: 1, SNP: 151000 "
## [1] "Genotype file: 1, SNP: 152000 "
## [1] "Genotype file: 1, SNP: 153000 "
## [1] "Genotype file: 1, SNP: 154000 "
## [1] "Genotype file: 1, SNP: 155000 "
## [1] "Genotype file: 1, SNP: 156000 "
## [1] "Genotype file: 1, SNP: 157000 "
## [1] "Genotype file: 1, SNP: 158000 "
## [1] "Genotype file: 1, SNP: 159000 "
## [1] "Genotype file: 1, SNP: 160000 "
## [1] "Genotype file: 1, SNP: 161000 "
## [1] "Genotype file: 1, SNP: 162000 "
## [1] "Genotype file: 1, SNP: 163000 "
## [1] "1 of 1 -- Vg= 1.1662 VE= 0.6517 -2LL= 1223.85   Clustering= average   Group number= 354   Group
## [1] "----------------------Sandwich bottom bun--------------------------------"
## [1] "Compression"
##      Type   Cluster   Group REML               VA
## [1,] "Mean" "average" "354" "1223.84830771592" "1.16624328973446"
##      VE
## [1,] "0.651675662671842"
```

```
## [1] "--------------------Final results presentations-----------------------"
## [1] "Generating summary"
## [1] "Genomic Breeding Values (GBV) ..."
## [1] "GAPIT.GS accomplished successfully!"
## [1] "Writing GBV and Acc..."
## [1] "GBV and accuracy distribution..."
## [1] "Compression portfolios..."
## [1] "Compression Visualization done"
## [1] "p3d objects transfered"
## [1] "Merge BLUP and BLUE"
## [1] "GAPIT before BLUP and BLUE"
## [1] "GAPIT after BLUP and BLUE"
## [1] "mlmm.adj_Proteincontent has been analyzed successfully!"
## [1] "The results are saved in the directory of  /home/marie.pegard/progeno/WorkShop"
## [1] "before ending GAPIT.Main"
## [1] "GAPIT accomplished successfully for multiple traits. Result are saved"
## [1] "It is OK to see this: 'There were 50 or more warnings (use warnings() to see the first 50)'"
```

```
pvals = -log10(myGAPIT_MLMM_proteincontent$GWAS$P.value)

res = cbind(Map[, 1:3], data.frame(pval = pvals[match(Map$SNP,
    myGAPIT_MLMM_proteincontent$GWAS$SNP)]))

plot_breaks = Map %>% group_by(Chr) %>% dplyr::summarise(medid = median(id))

ggplot(res, aes(x = 1:nrow(Map), y = pval, color = as.factor(Chr))) +
    geom_point(show.legend = F) + scale_color_manual(values = rep_len(magma(11,
    alpha = 0.7)[-11], length.out = length(unique(Map$Chr)))) +
    geom_hline(yintercept = 6.5) + scale_x_continuous(name = "Chromosomes",
    breaks = plot_breaks$medid, labels = plot_breaks$Chr) +
    theme_pubclean()
```



Seed yield

We are going to do the same thing for the Seedyield.

```
data_gwas = na.omit(Adj_PhenoSoy[match(rownames(GV),
    Adj_PhenoSoy$EUCLEGID), c("EUCLEGID", "adj_Seedyield")])

K = GV[match(data_gwas$EUCLEGID, rownames(GV)), match(data_gwas$EUCLEGID,
    colnames(GV))]
dim(data_gwas)
```

```
## [1] 357   2
```

```
dim(K)
```

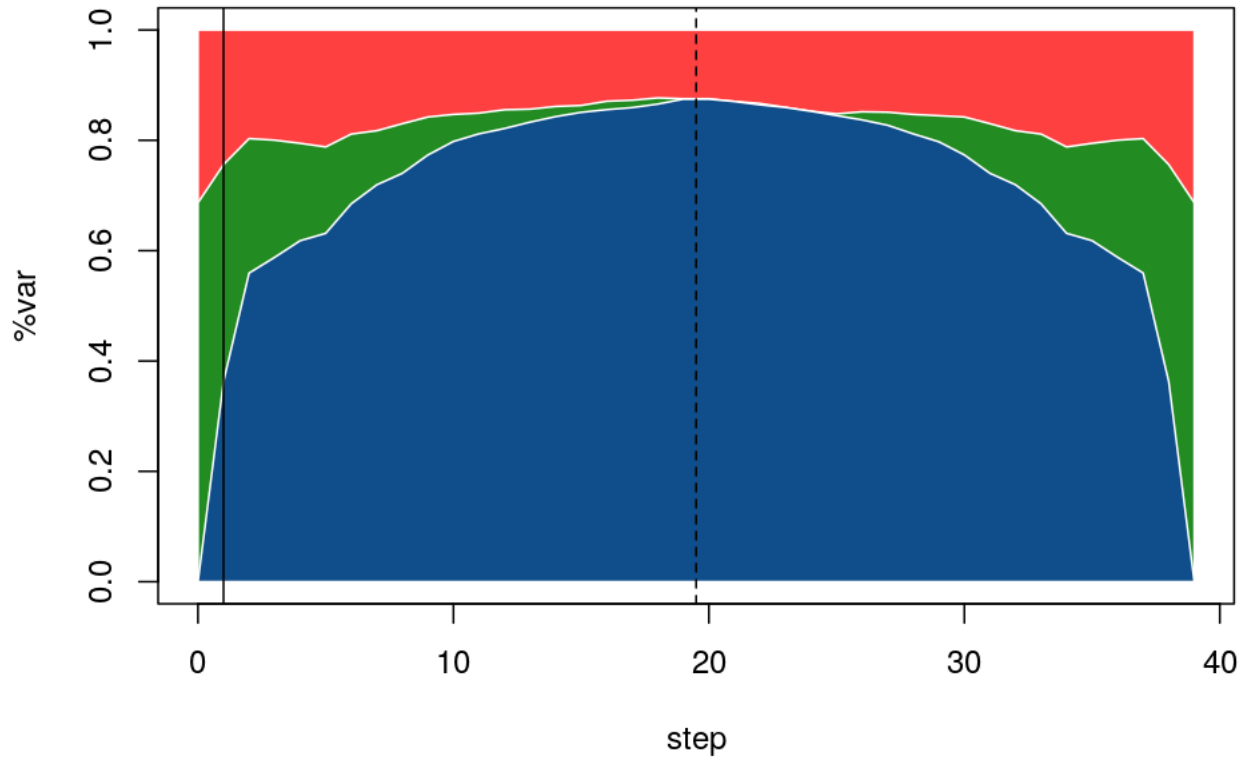
```
## [1] 357 357
```

```
C = M[match(data_gwas$EUCLEGID, rownames(M)), ] + 1

mygwas_Seedyield <- mlmm(Y = data_gwas$adj_Seedyield,
    X = C, K = K, nbchunks = 2, maxsteps = 20)
```

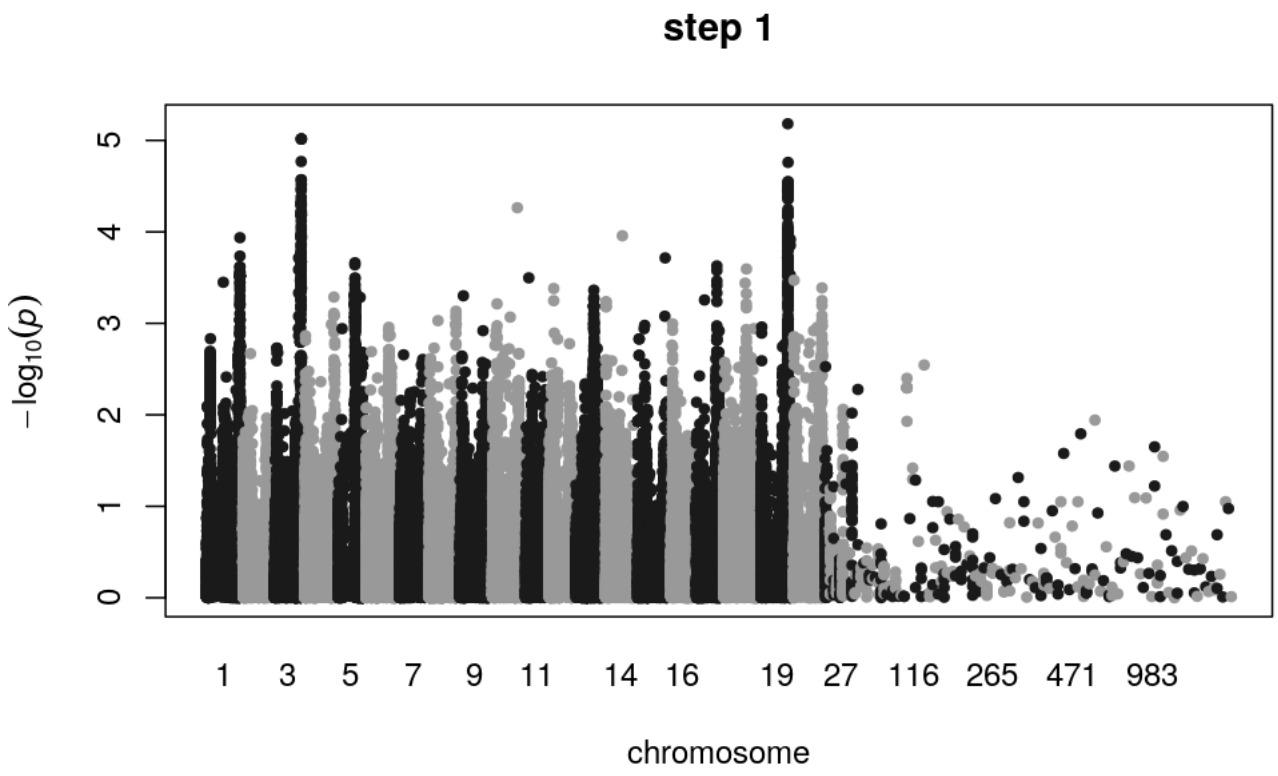
```
## null model done! pseudo-h= 0.687
## step 1 done! pseudo-h= 0.617
## step  2  done! pseudo-h= 0.553
## step  3  done! pseudo-h= 0.515
## step  4  done! pseudo-h= 0.462
```

```
## step  5  done! pseudo-h= 0.425
## step  6  done! pseudo-h= 0.401
## step  7  done! pseudo-h= 0.349
## step  8  done! pseudo-h= 0.346
## step  9  done! pseudo-h= 0.304
## step  10  done! pseudo-h= 0.242
## step  11  done! pseudo-h= 0.199
## step  12  done! pseudo-h= 0.19
## step  13  done! pseudo-h= 0.141
## step  14  done! pseudo-h= 0.117
## step  15  done! pseudo-h= 0.084
## step  16  done! pseudo-h= 0.105
## step  17  done! pseudo-h= 0.095
## step  18  done! pseudo-h= 0.083
## step  19  done! pseudo-h= 0.006
## backward analysis
## creating output
```

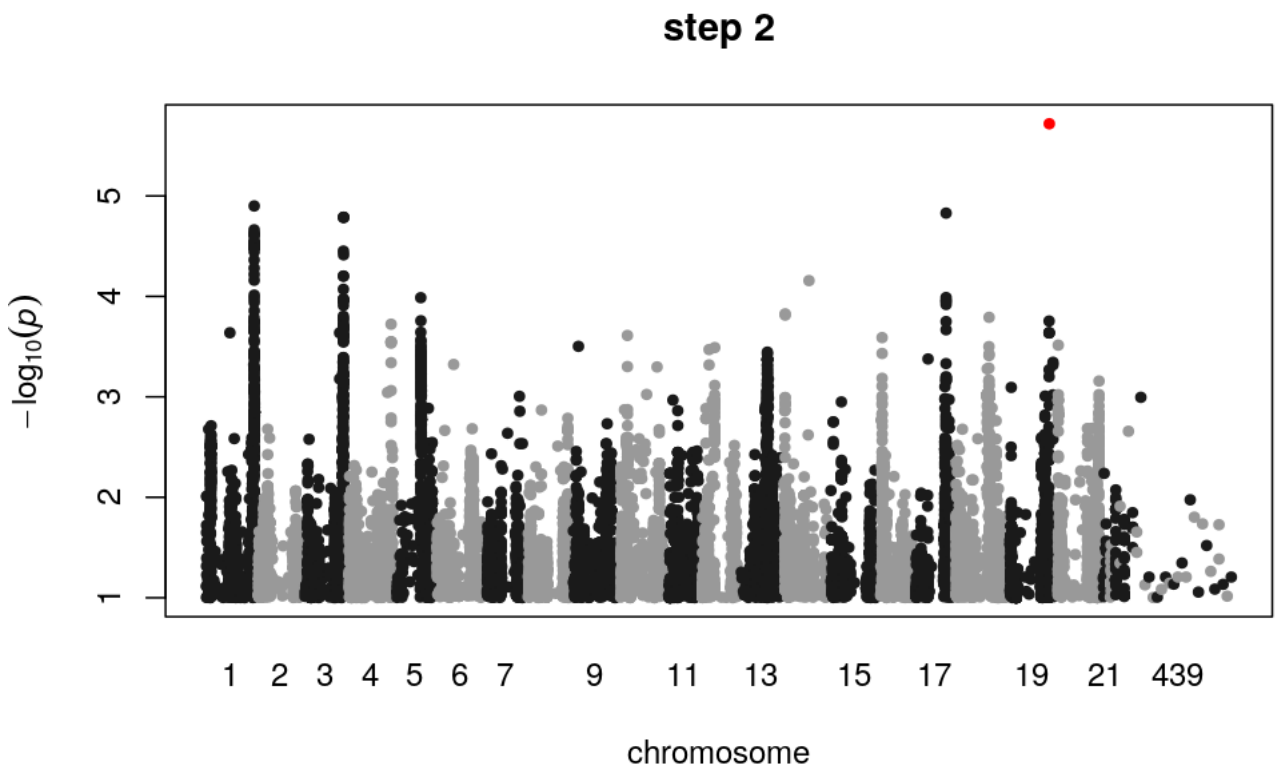
```r
plot_step_RSS(mygwas_Seedyield)  # % variance plot
abline(v = which.min(mygwas_Seedyield$step_table$extBIC),
    col = "Black", cex = 3)
```



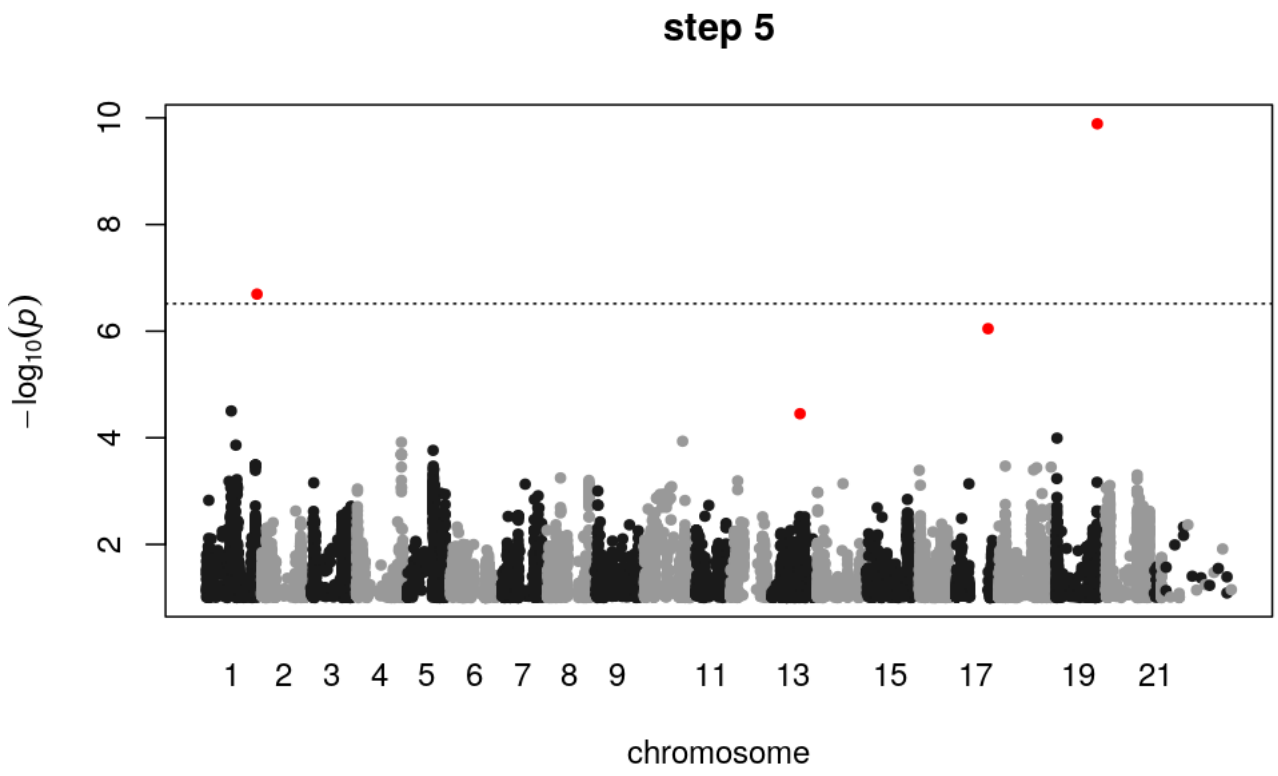

```r
plot_fwd_GWAS(mygwas_Seedyield, 1, Map[, 1:3], 1, main = "step 1")  # 1st mlmm step plot
```



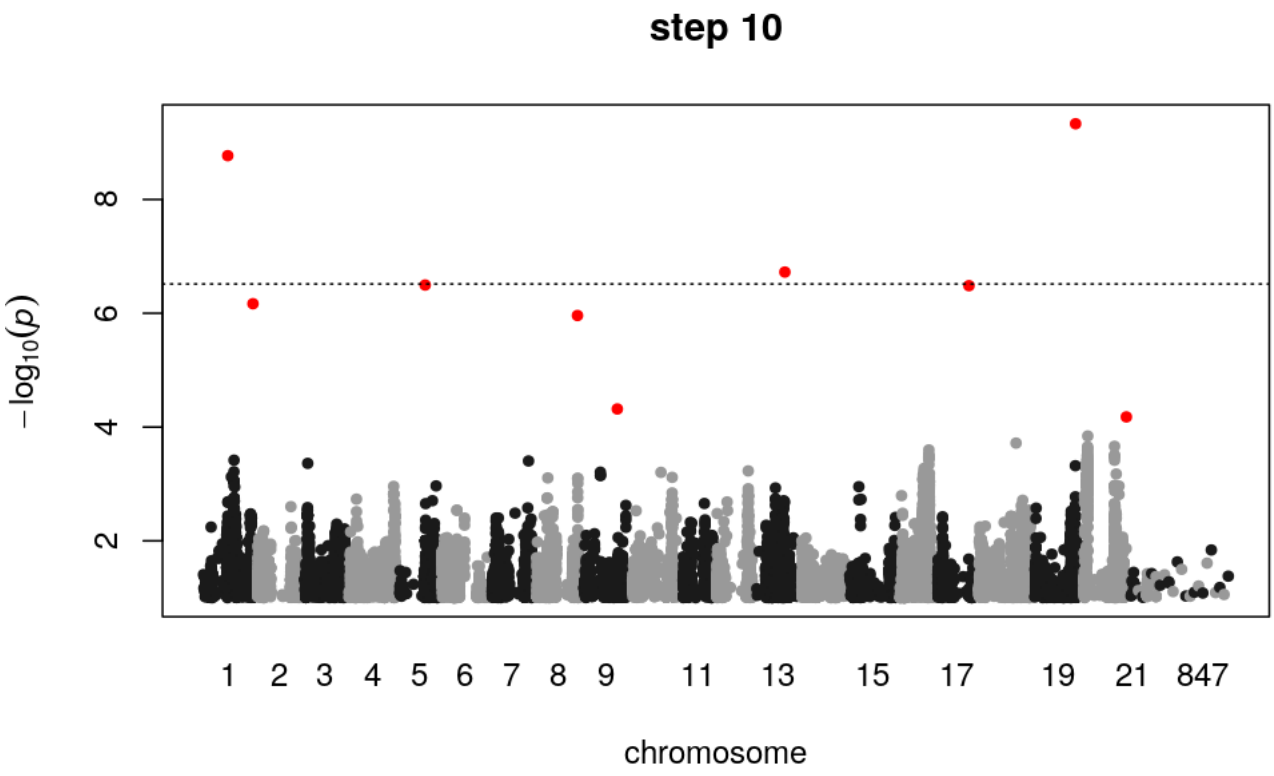

```
plot_fwd_GWAS(mygwas_Seedyield, 2, Map[, 1:3], 0.1,
    main = "step 2")  # 2nd mlmm step plot
```



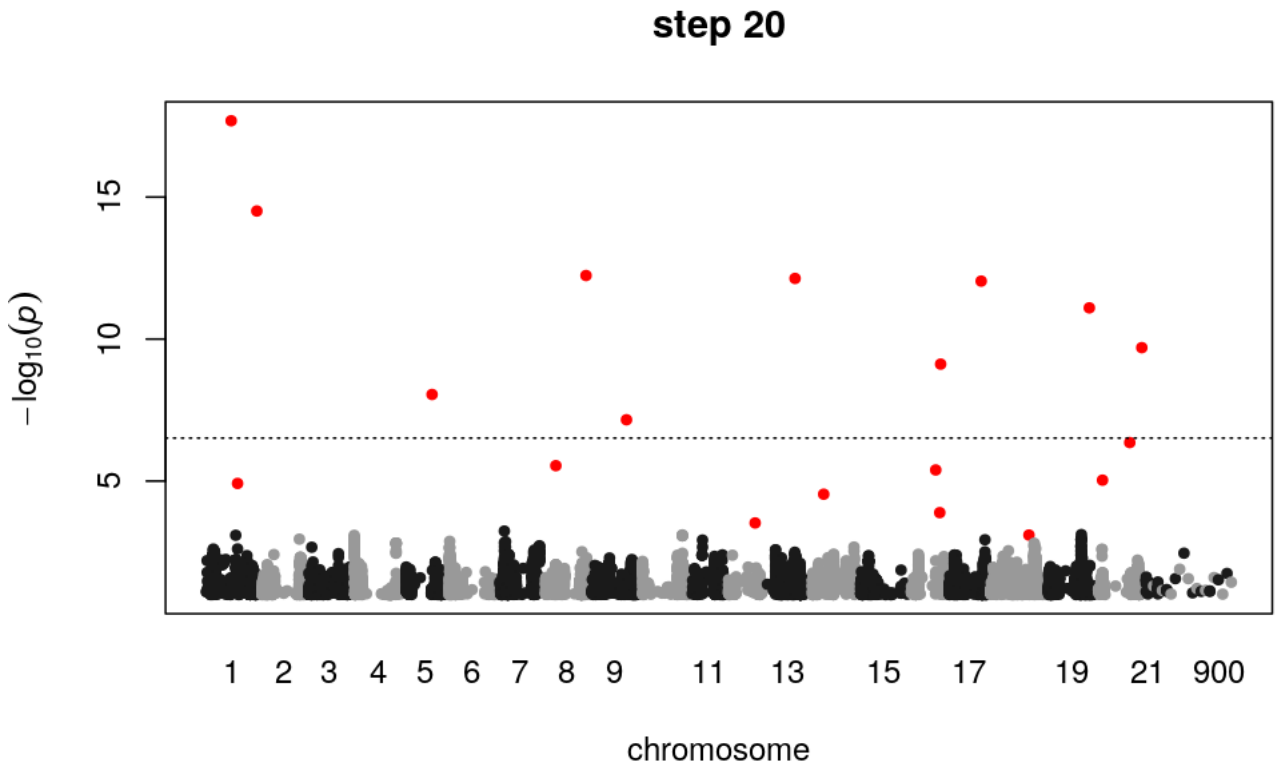

```
plot_fwd_GWAS(mygwas_Seedyield, 5, Map[, 1:3], 0.1,
    main = "step 5")  # 3rd mlmm step plot
```



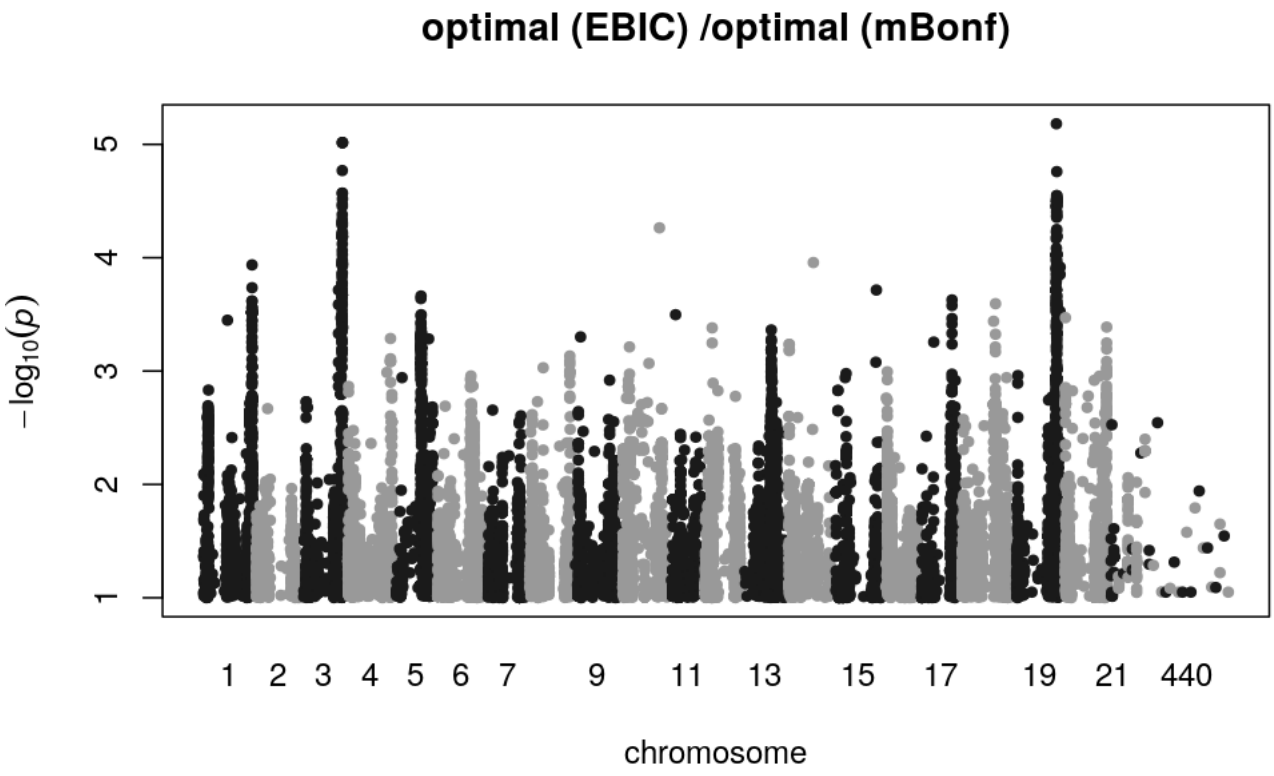

```
plot_fwd_GWAS(mygwas_Seedyield, 10, Map[, 1:3], 0.1,
    main = "step 10")  # 3rd mlmm step plot
```




```
plot_fwd_GWAS(mygwas_Seedyield, 20, Map[, 1:3], 0.1,
    main = "step 20")  # 3rd mlmm step plot
```




```
plot_opt_GWAS(mygwas_Seedyield, "extBIC", Map[, 1:3],
    0.1, main = "optimal (EBIC) /optimal (mBonf)")
```




```
# optimal step according to eBIC plot
```

In the case of grain yield, if the Bonferonni threshold or the BIC criterion is used, no QTL is detected.

Genomic Selection (Extract from Pégard 2018)

Genomic selection (GS) is the direct descendant of marker-assisted selection (SAM) applied to quantitative traits (Meuwissen et al., 2001). Its application is based on dense genotyping all along the genome and the construction of predictive models using statistical methods, usually a derivative of a mixed model BLUP or a Bayesian method. These models simultaneously rely on the information provided by all markers to estimate the value of individuals that are candidates for selection. With adapted models, GS is able to predict the total genotypic value by estimating the additive (GEBV : genomic breeding value), dominance and epistatic part present in the phenotype of an individual / population. Markers are generally distributed over the entire genome and for the majority of them there is no a priori information indicating whether or not they are related to a QTL. The predictive model is calibrated with a set of individuals that have been phenotyped (or with known estimated additive value) and genotyped. These individuals constitute the calibration (training) population. Candidates for selection are evaluated and selected according to their genotyping information using previously constructed predictive models. Candidates may be progenies of the training population or individuals from other somehow related populations.

Accuracy of genomic selection

As with conventional selection, the accuracy of genomic prediction is estimated with the correlation between the true breeding value (TBV) and its estimate (GEBV : Genomic Estimated Breeding Value). When using

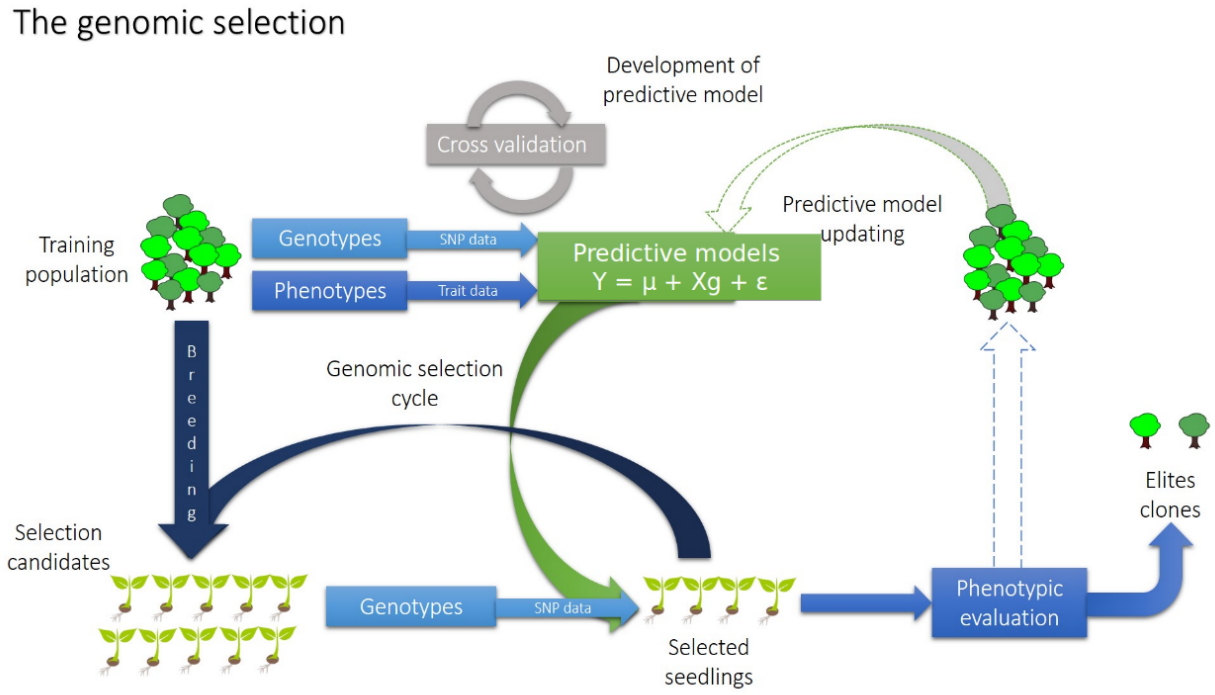


Figure 3: Concept of genomic selection adapted from Grattapaglia (2014) to the Poplar context extract from Pégard 2018

empirical data, the TBV of an individual is unknown, and accuracy must be estimated by cross-validation. This consists in splitting the training population into several subsamples. Each subsample is successively used as a validation population, and its GEBVs predicted by a model that is calibrated with the other subsamples. All subsamples participate one single time as validation. The correlation between GEBVs and individual phenotypes is then calculated for each validation population. It is expected that residuals in the phenotypes are uncorrelated to breeding values, thus the correlation of GEBV and phenotypes can be considered as a good proxy of the correlation between TBV and GEBV. The calculated correlation is called predictive ability or prediction accuracy depending on the authors. It measures the ability of genomic selection to predict observed values, rather than the true additive value. If the observed values are phenotypes, the accuracy of prediction can be inferred by dividing the predictive ability with the square root of the heritability of the trait in question. ($r_{GEBV,TBV} = r_{GEBV,Phenotypes}/\sqrt{h^2}$) (Falconer, 1981).

The accuracy of genomic prediction is affected by several factors : the relatedness between training and test populations, the number of individuals in the training population, the linkage disequilibrium between markers and QTLs, the statistical method used to estimate GEBVs, the heritability of the trait, the molecular marker density, the type of markers and the genetic architecture of the trait (number of genes and distribution of their effects) (Hayes et al., 2009a; Jannink et al., 2010; Lorenz et al., 2011; Grattapaglia, 2014).

The predicting ability (or accuracy) is the most common criteria to estimate the genomic prediction performance (Daetwyler et al., 2013). Other complementary criteria for prediction quality are also proposed, like the slope and intercept of the regression of phenotypes on prediction. This slope should be close to 1 and the intercept close to zero. There are many reasons for deviations in slope and intercept, like model‘s deficiencies as wrong variance partition or incomplete model, this induced systematic biases or nonrandom choice of individuals for training and validation population (Patry and Ducrocq, 2011a,b; Mäntysaari et al., 2010).

GBLUP

Among the different statistical methods to obtain GEBVs, two main groups can be distinguished : approaches that estimate an additive effect associated with each marker and approaches that directly give the additive value of individuals.

Since the number of markers (p) is often greater than the number of individuals (n) in the calibration model, the known p>>n problem, estimating the effect of markers by multiple regression using ordinary least squares is not possible. Some kind of variable selection or shrinkage estimation procedure is then required to make such problem solvable (De Los Campos et al., 2009). In recent years, several methods have been developed : derivatives of BLUP methods (Henderson, 1975), Bayesian methods and non-parametric methods (Gianola and van Kaam, 2008; Neves et al., 2012; González-Camacho et al., 2012; Ornella et al., 2014; Howard et al., 2014).

One of the simplest formulas for the estimation of GEBV is the following : $y = \mu + Za + \epsilon$

where y is the vector of observations (n×1), μ the mean of the observations, a the vector of markers' effects (p × 1), Z an incidence matrix (n × p) containing the copy number (0, 1 or 2) of the most frequent allele and ϵ the vector of residuals (n×1), with n individuals (observations) and p markers (unknowns or parameters). GEBV of individuals can then be obtained as a result of summing up across all markers their respective additive effects in a.

For approaches giving GEBV directly (GBLUP), a basic formula is : $y = \mu + Xg + \epsilon$

where g is a vector (p × 1) of GEBV, and X a design matrix linking observations to individuals. GEBVs follow a normal distribution, with a covariance between effects modelled through an additive relationship matrix that is derived from markers for GEBV or from pedigree in classical pedigree-based BLUPs.

A wide variety of statistical methods have been developed to handle the problem of greater number of parameters than observations by following different strategies. Roughly, the different strategies are thought to accommodate implicitly different underlying genetic a architectures of quantitative traits, from simpler

infinitesimal-like architectures to those more complex with a variety of heterogeneous effects across genes. Thus, some statistical methods for GS are based on a constraint that imposses the same variance for all markers (RR-BLUP, GBLUP, etc.), and that corresponds to pseudo-infinitesimal genetic architectures (very large number of genes with very small effects). Other methods impose variable selection and/or shrinkage strategies to reduce the number of relevant parameters to be estimated, and this can be done for instance by using a priori information in a Bayesian framework that forces most effects to be close to zero with just a few with larger values. With the Bayes family of methods (A, B, etc) there is the possibility to assume variances that are specific for each marker. This allows greater emphasis to be placed on certain markers with respect to others that will end up with negligible effects. Another somehow simpler alternative is to use an iterative weighted GBLUP (Legarra et al., 2009). Each marker is weighted depending on its own estimated effect, obtained in a previous iteration of GBLUP and the process can be repeated several cycles to narrow the set of selected markers. Some implementations showed similar results to Bayesian methods, with the advantage of being much faster and easier (WANG et al., 2012).

In this workshop, we will test two methods, the single-stepGBLUP (GBLUP where genotyped and non-genotyped individuals are combined) and the QBLUP (GBLUP with the addition of QTLs in fixed or random effects). Bayesian methods are not tested here, as they are often more effective with oligogenic rather than infinitesimal traits. Moreover, they are often longer to implement and require to check the convergence of the model (several tens of thousands of iterations).

Practical

Cross- validation and Test Set

As explained above, the implementation of a cross-validation allows the performance of the model to be estimated. To do this, the data set is first divided into two parts. 20% of the data set is defined as an independent test set. The cross-validation will be carried out in the remaining 80%. This part will itself be divided in two with variable proportions, one part will be used to train the model and the second part to validate the model. Several iterations are performed. The test set will serve as verification and will be common to all the cross validations.

Training Single trait

The first step is to define the individuals who will be used to train the model and those who will serve as validation. To ensure that the model is not over fitted, an independent data set (test) will be created to serve as a common validation for all cross-validations.

The independent data set will consist of 20% of the data. The remaining 80% of the data will be used for cross-validation. Three training population sizes will be tested. The model will be trained with 20% of the data, 50% of the data and 70% of the data.

```
# I order the matching matrix according to the
# genotyping data.
HX2 = HX[match(Adj_PhenoSoy$EUCLEGID, rownames(HX)),
    match(Adj_PhenoSoy$EUCLEGID, colnames(HX))]

# Total number of individual
nbtot = nrow(HX2)

nbtestset = nbtot * 0.2  # 20% of the individuals as Test set
nbval = (nbtot - nbtestset) * 0.5  # 80% of the remainin

# Sampling the testset among the individuals
ind_test = sample(Adj_PhenoSoy$EUCLEGID, nbtestset)
```

Figure 4: Cutting of the data set in Training, Validation and Test part.

```
# Creation of incidence matrice.
inc.H <- matrix(0, nrow(Adj_PhenoSoy), ncol = ncol(HX2))
colnames(inc.H) <- colnames(HX2)
rownames(inc.H) <- Adj_PhenoSoy$EUCLEGID

for (i in colnames(inc.H)) {
    inc.H[which(rownames(inc.H) == i), i] <- 1
}


# Individual sampling for the cross-validation, 50%
# of the individual are training the model and the
# 50% relaing are used to validate the model
INDVAL50 = list()
for (i in 1:10) {
    ind_val = sample(rownames(HX2)[!rownames(HX2) %in%
        ind_test], nbval)

    INDVAL50[[i]] = ind_val

}

# 20% of the individual are training the model and
# the 80% relaing are used to validate the model

nbval = (nbtot - nbtestset) * 0.8
```

```
INDVAL20 = list()
for (i in 1:10) {
    ind_val = sample(rownames(HX2)[!rownames(HX2) %in%
        ind_test], nbval)

    INDVAL20[[i]] = ind_val

}

# 70% of the individual are training the model and
# the 30% relaing are used to validate the model

nbval = (nbtot - nbtestset) * 0.3

INDVAL70 = list()
for (i in 1:10) {
    ind_val = sample(rownames(HX2)[!rownames(HX2) %in%
        ind_test], nbval)

    INDVAL70[[i]] = ind_val

}
```

First we do the cross-validation with 50% of the data to train the model. Then the others. In all cases the principle is the same, only the individuals change.

The results are stored in a table for later comparison.

```
Results = data.frame(Trait = c("adj_Proteincontent",
    "adj_Seedyield"), Model = "ST", TrainingPerc = 50,
    rep = 1:10, heritability = NA, AIC = NA, cor_val = NA,
    cor_test = NA, rank_val = NA, rank_test = NA, Acc_val = NA,
    Acc_test = NA, slope_val = NA, slope_test = NA)



for (i in 1:10) {
    ind_val = INDVAL50[[i]]  # individual as validation

    phenotp = Adj_PhenoSoy  # temporary phenotypic dataset
    # Validation and Test individuals are masked to be
    # predicted by the model with the genomic
    # information

    phenotp$adj_Proteincontent[phenotp$EUCLEGID %in%
        c(ind_test, ind_val)] = NA

    # GBLUP
    GBLUP <- remlf90(fixed = adj_Proteincontent ~ 1,
        generic = list(genetic = list(inc.H, HX2)),
        data = phenotp)

    # we are extractin GEBV (Genomic Estimated Breeding
    # Value)
    gebv = GBLUP$ranef$generic_genetic[[1]]$value
```

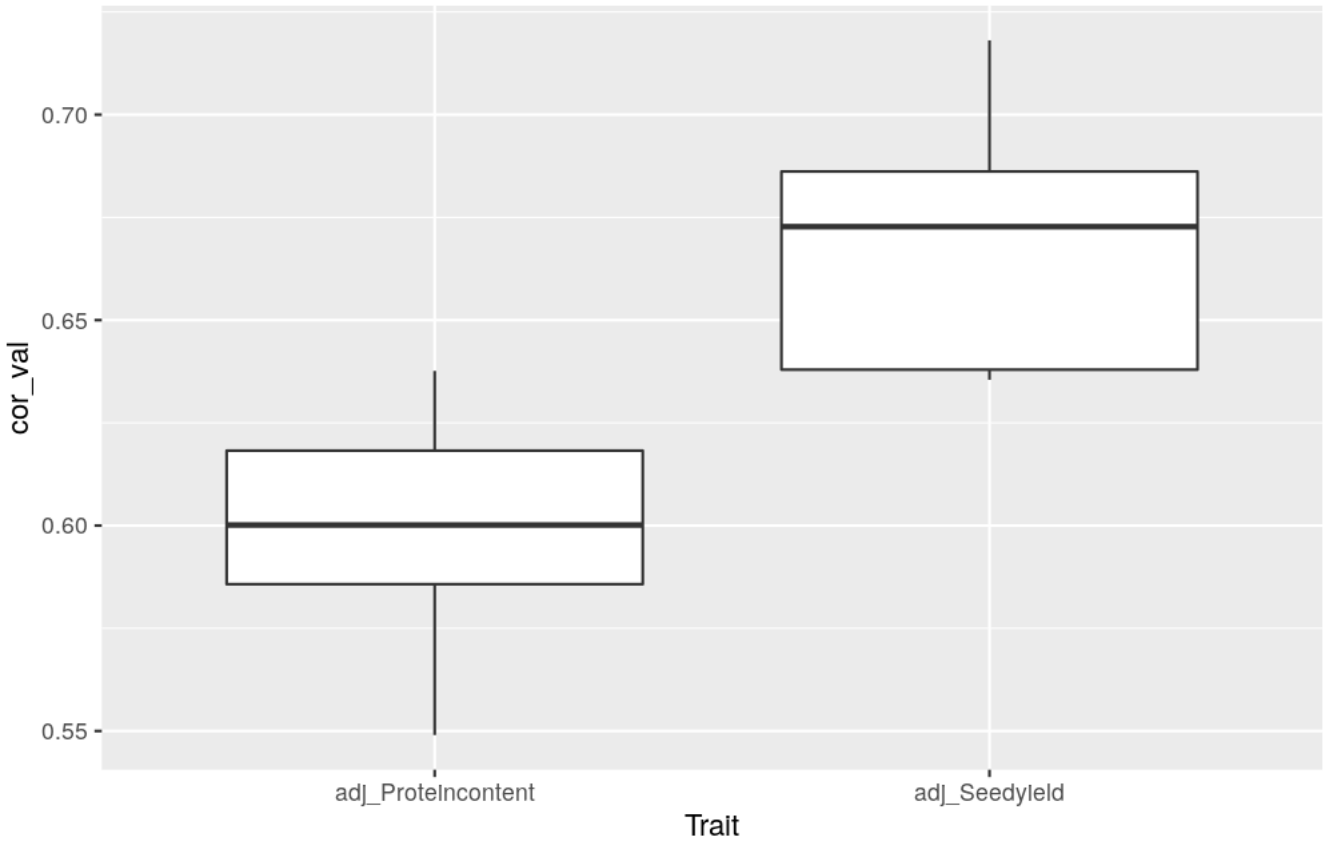
```
    # I put the individuals name to compare the results
    names(gebv) = Adj_PhenoSoy$EUCLEGID

    # i stock in the dataset the predicted gebv
    Adj_PhenoSoy$gebv = gebv

    # Estimation of the heritability
    Results$heritability[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = GBLUP$var["generic_genetic",
        "Estimated variances"]/sum(GBLUP$var[, "Estimated variances"])

    # Model AIC extraction
    Results$AIC[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = GBLUP$fit$AIC

    # Estimation of the Predictive ability in the
    # validation set
    Results$cor_val[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], use = "na.or.complete")

    # Estimation of the Predictive ability in the
    # Testset
    Results$cor_test[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], use = "na.or.complete")

    # Estimation of the Rank prediction in the
    # validation set
    Results$rank_val[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], use = "na.or.complete", method = "spearman")

    # Estimation of the Rank prediction in the Test set
    Results$rank_test[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], use = "na.or.complete", method = "spearman")

    # Estimation of the Accuracy in the validation set
    Results$Acc_val[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], use = "na.or.complete")/sqrt(heritabilities["Proteincontent"])

    # Estimation of the Accuracy in the Testset
    Results$Acc_test[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], use = "na.or.complete")/sqrt(heritabilities["Proteincontent"])
```

```
# Determination of the slope between phenotypes and
# gebv in the validation set
Results$slope_val[Results$Trait == "adj_Proteincontent" &
    Results$rep == i] = coef(lm(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
    ind_val] ~ Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_val]))[2]

# Determination of the slope between phenotypes and
# gebv in the Test set
Results$slope_test[Results$Trait == "adj_Proteincontent" &
    Results$rep == i] = coef(lm(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
    ind_test] ~ Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_test]))[2]

# We are doing the same for Seed yield
phenotp = Adj_PhenoSoy  # temporary phenotypic dataset
# Validation and Test individuals are masked to be
# predicted by the model with the genomic
# information

phenotp$adj_Seedyield[phenotp$EUCLEGID %in% c(ind_test,
    ind_val)] = NA

# GBLUP
GBLUP <- remlf90(fixed = adj_Seedyield ~ 1, generic = list(genetic = list(inc.H,
    HX2)), data = phenotp)

# we are extractin GEBV (Genomic Estimated Breeding
# Value)
gebv = GBLUP$ranef$generic_genetic[[1]]$value
# I put the individuals name to compare the results
names(gebv) = Adj_PhenoSoy$EUCLEGID

# i stock in the dataset the predicted gebv
Adj_PhenoSoy$gebv = gebv

# Estimation of the heritability
Results$heritability[Results$Trait == "adj_Seedyield" &
    Results$rep == i] = GBLUP$var["generic_genetic",
    "Estimated variances"]/sum(GBLUP$var[, "Estimated variances"])

# Model AIC extraction
Results$AIC[Results$Trait == "adj_Seedyield" &
    Results$rep == i] = GBLUP$fit$AIC

# Estimation of the Predictive ability in the
# validation set
Results$cor_val[Results$Trait == "adj_Seedyield" &
    Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
    ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_val], use = "na.or.complete")

# Estimation of the Predictive ability in the
```

```
    # Testset
    Results$cor_test[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], use = "na.or.complete")

    # Estimation of the Rank prediction in the
    # validation set
    Results$rank_val[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], use = "na.or.complete", method = "spearman")

    # Estimation of the Rank prediction in the Test set
    Results$rank_test[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], use = "na.or.complete", method = "spearman")

    # Estimation of the Accuracy in the validation set
    Results$Acc_val[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], use = "na.or.complete")/sqrt(heritabilities["Seedyield"])

    # Estimation of the Accuracy in the Testset
    Results$Acc_test[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], use = "na.or.complete")/sqrt(heritabilities["Seedyield"])

    # Determination of the slope between phenotypes and
    # gebv in the validation set
    Results$slope_val[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = coef(lm(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_val] ~ Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val]))[2]

    # Determination of the slope between phenotypes and
    # gebv in the Test set
    Results$slope_test[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = coef(lm(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_test] ~ Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test]))[2]


}
```

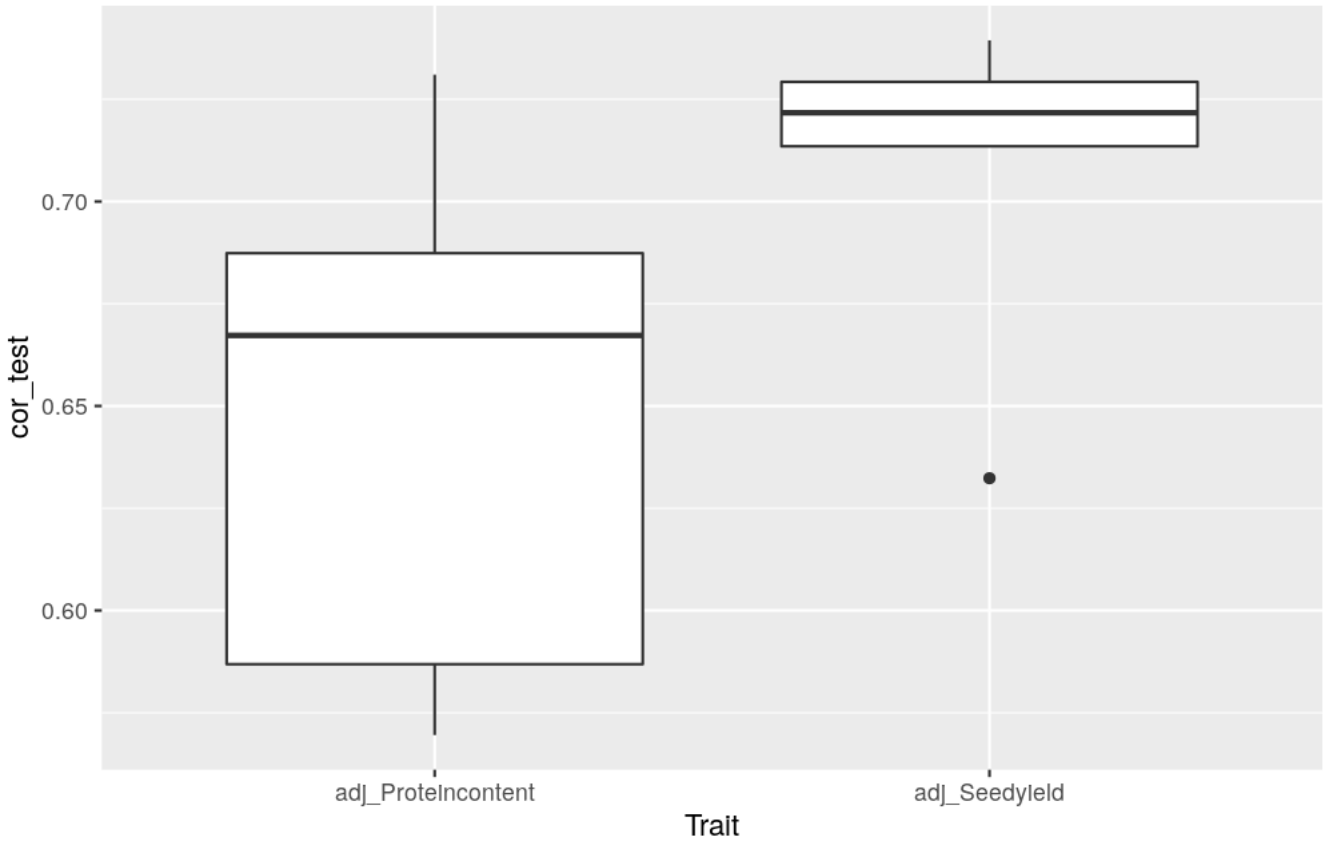
In a first step we observe the predictive ability (correlation) between the phenotype and the values predicted by the trained model with 50% of the data in both traits in the validation set. The predictive ability is hiher than 0.5 for both traits and the seedyield present the highest predictive ability.

```
ggplot(Results, aes(x = Trait, y = cor_val)) + geom_boxplot()
```



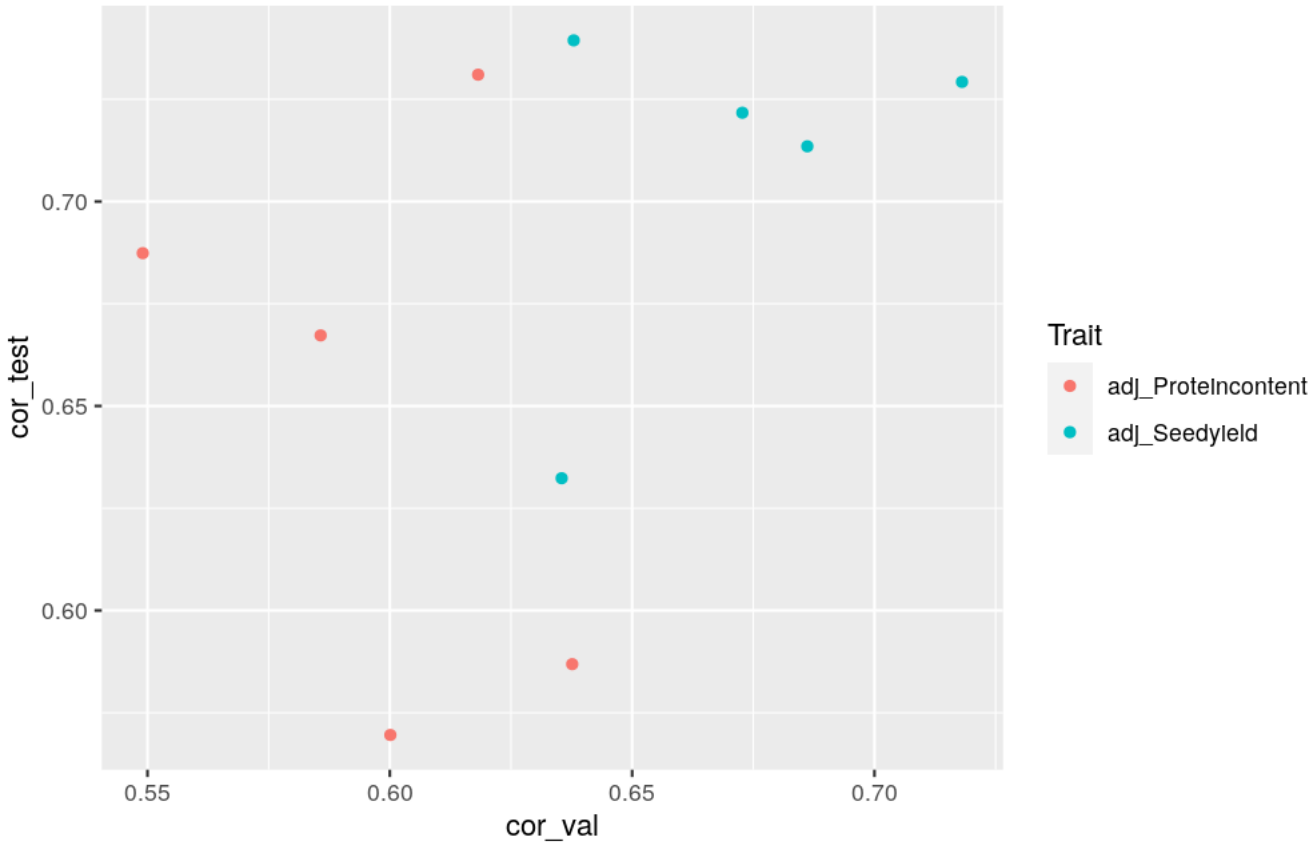
The same with the Set test. The values obtained are close to the previous ones, even if the difference between the two traits is reduced. Also, we can see an important variation between the repetitions for the yield. This means that certain combinations of individuals train the model better than others from the point of view of the Set test.

```
ggplot(Results, aes(x = Trait, y = cor_test)) + geom_boxplot()
```



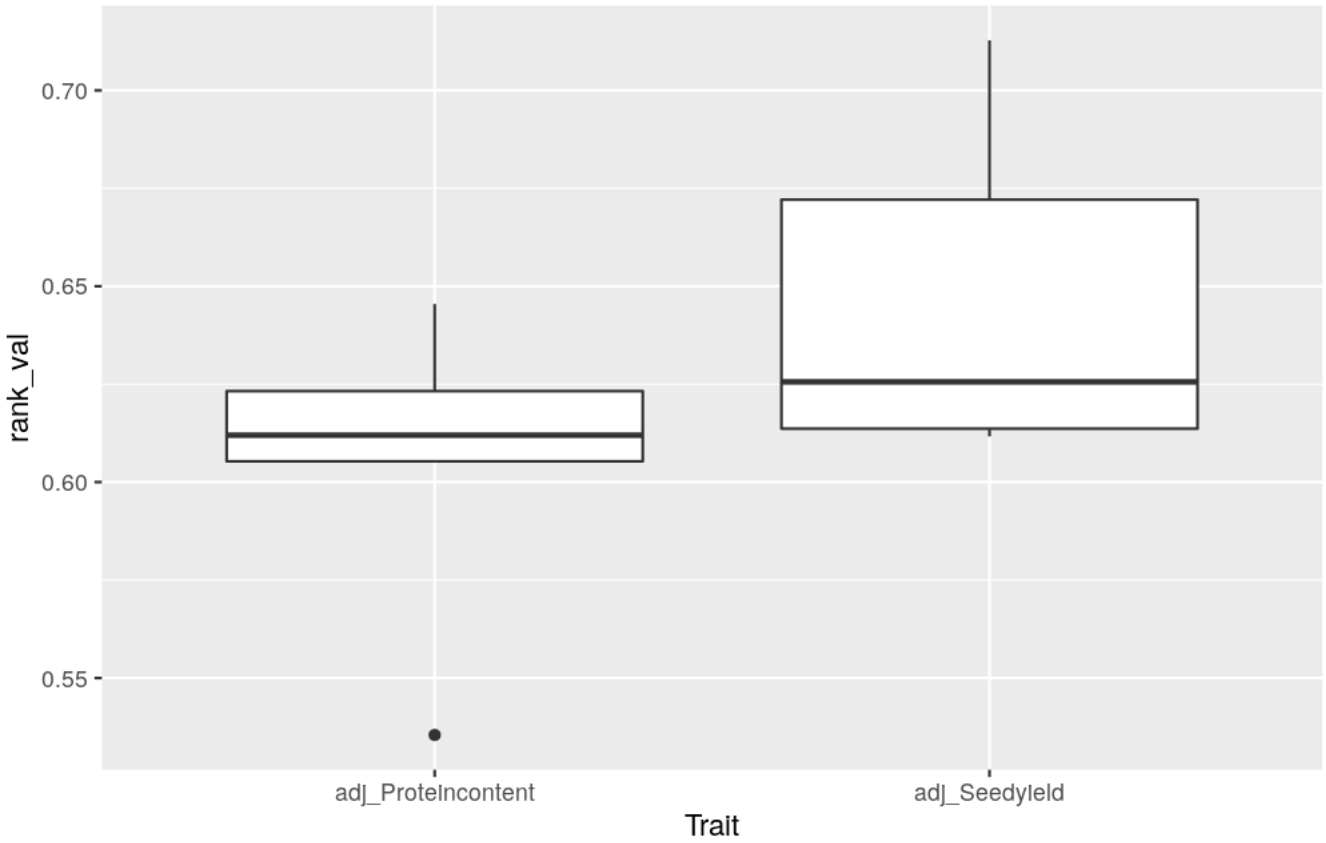
We can see on the following graph that the best combination of individuals to train the model for validation is not always the best one to predict the test set.

```
ggplot(Results, aes(x = cor_val, y = cor_test, color = Trait)) +
    geom_point()
```



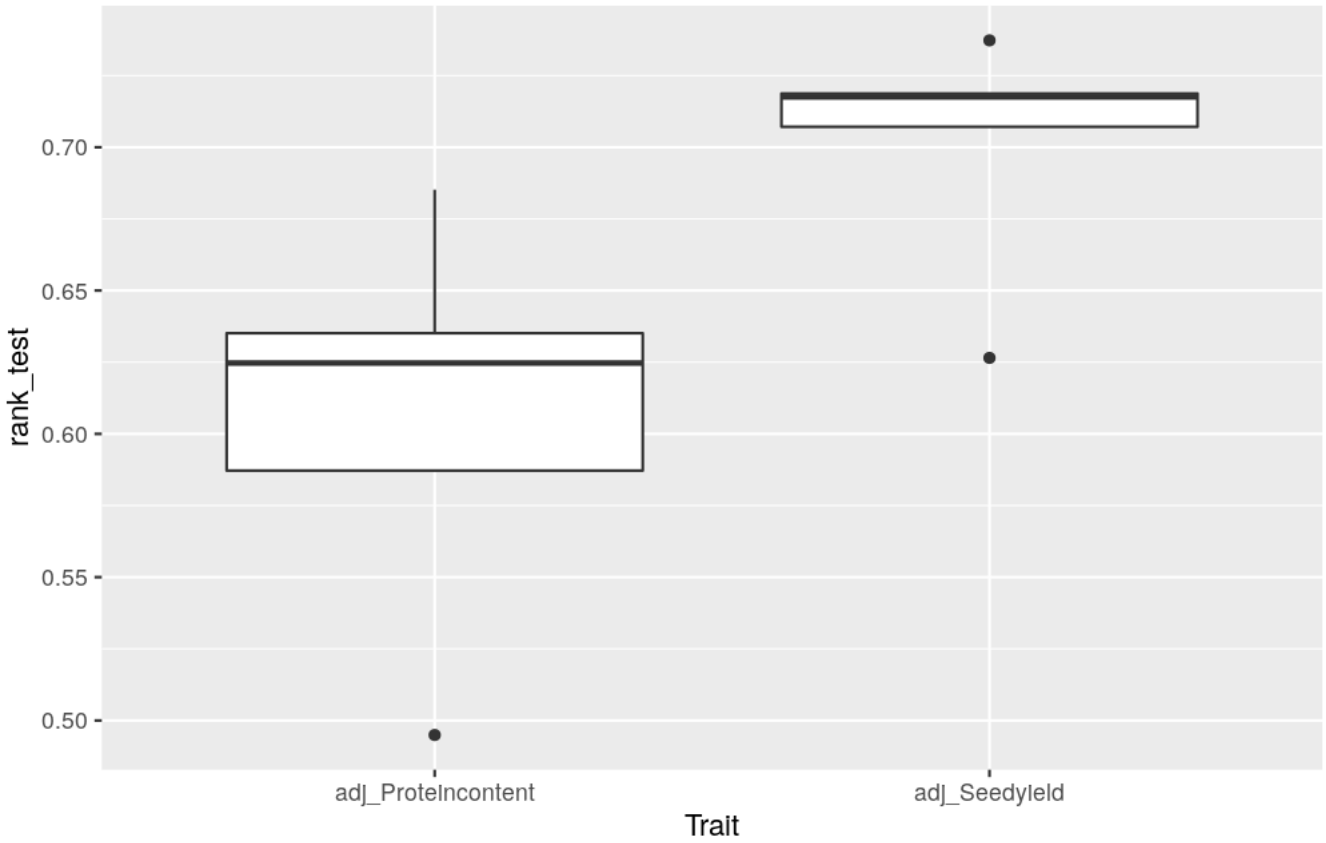
In some cases, we are more interested in selecting individuals based on their rank rather than their predicted value, for example, early or late heading individuals. Let's look at what the ranking of individuals gives. First in validation

```
ggplot(Results, aes(x = Trait, y = rank_val)) + geom_boxplot()
```



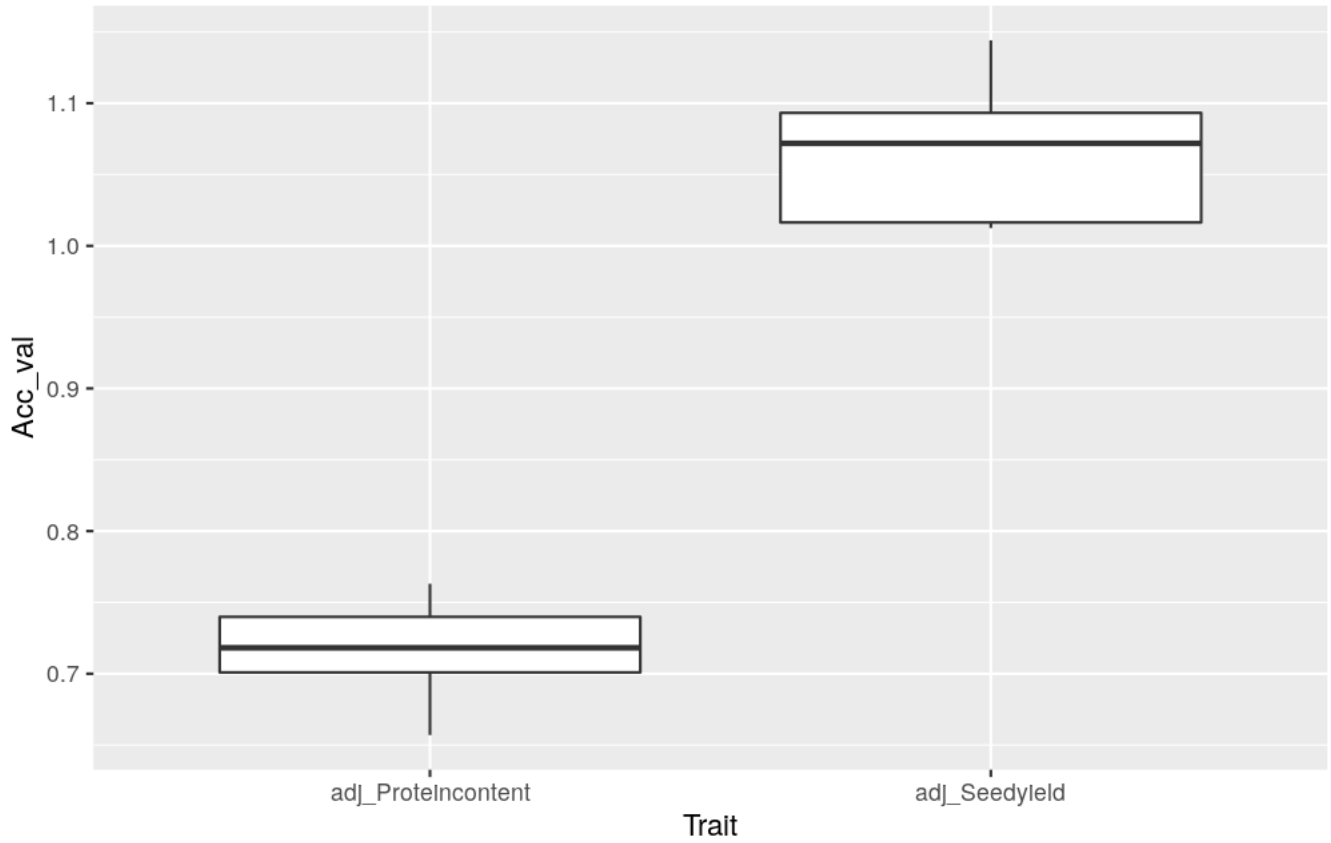
Then in Test Set

```
ggplot(Results, aes(x = Trait, y = rank_test)) + geom_boxplot()
```



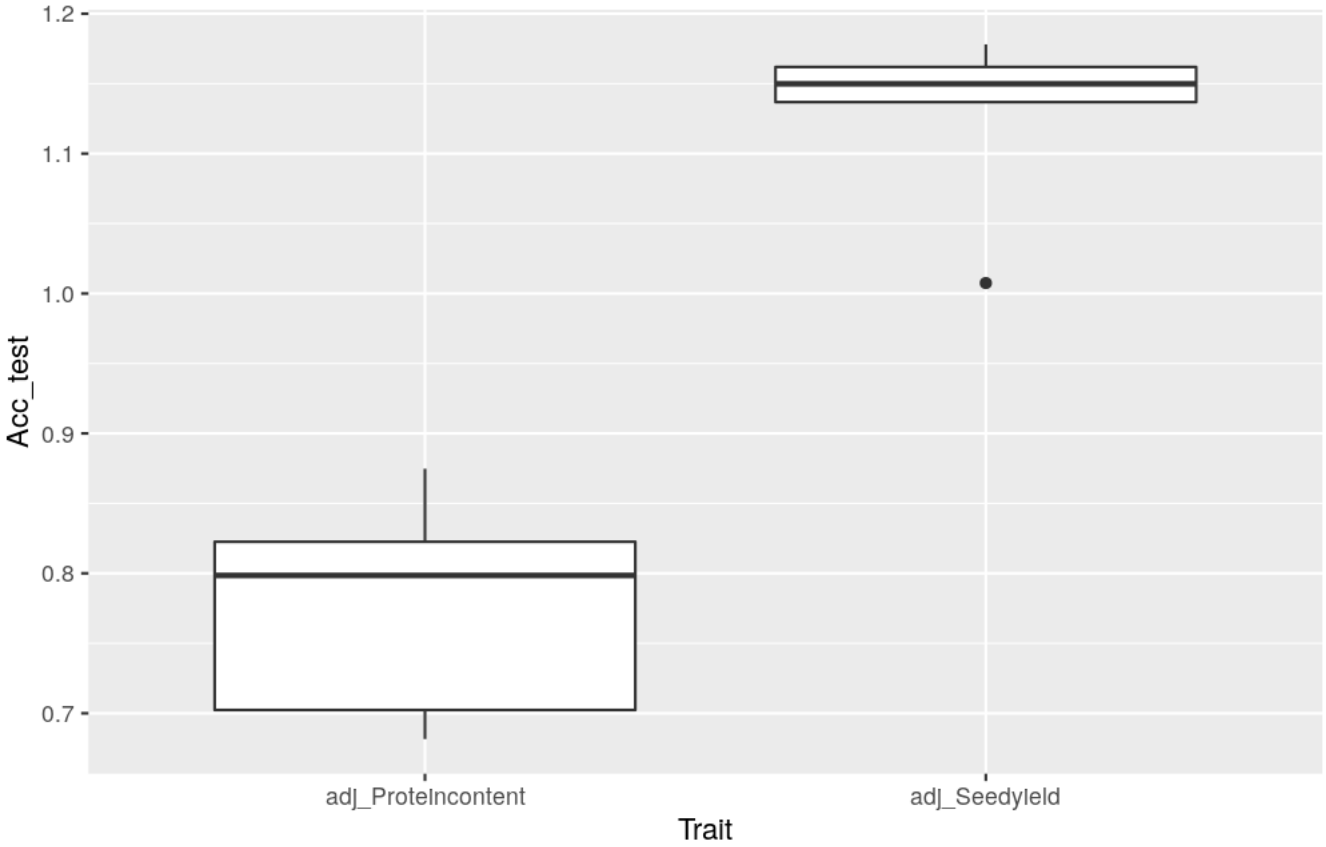
As we use phenotypes we do not have access to the TBV, it can be estimated via accuracy as explained above. First for validation. The closer the accuracy is to 1, the better the model performs.

```
ggplot(Results, aes(x = Trait, y = Acc_val)) + geom_boxplot()
```



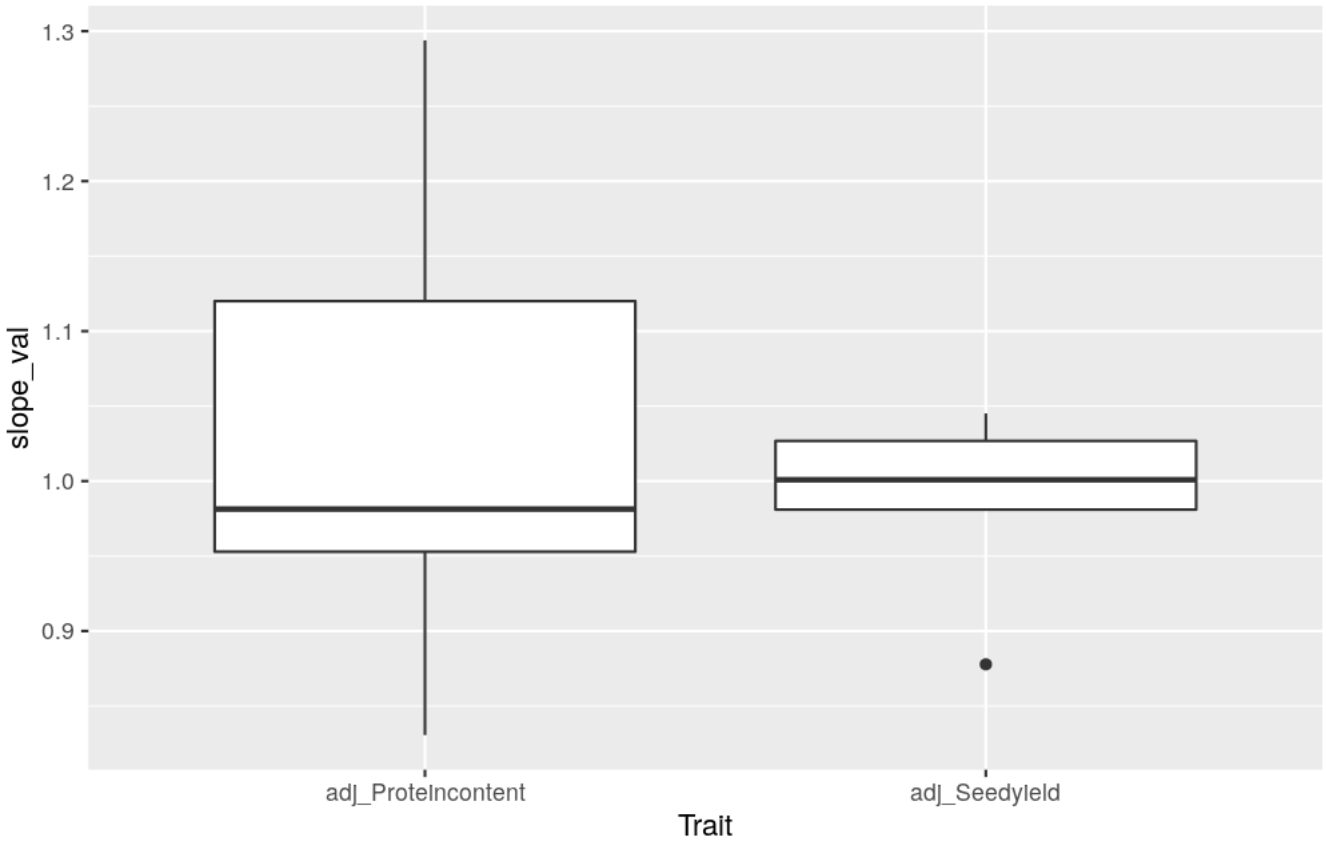
Then in Test Set

```
ggplot(Results, aes(x = Trait, y = Acc_test)) + geom_boxplot()
```



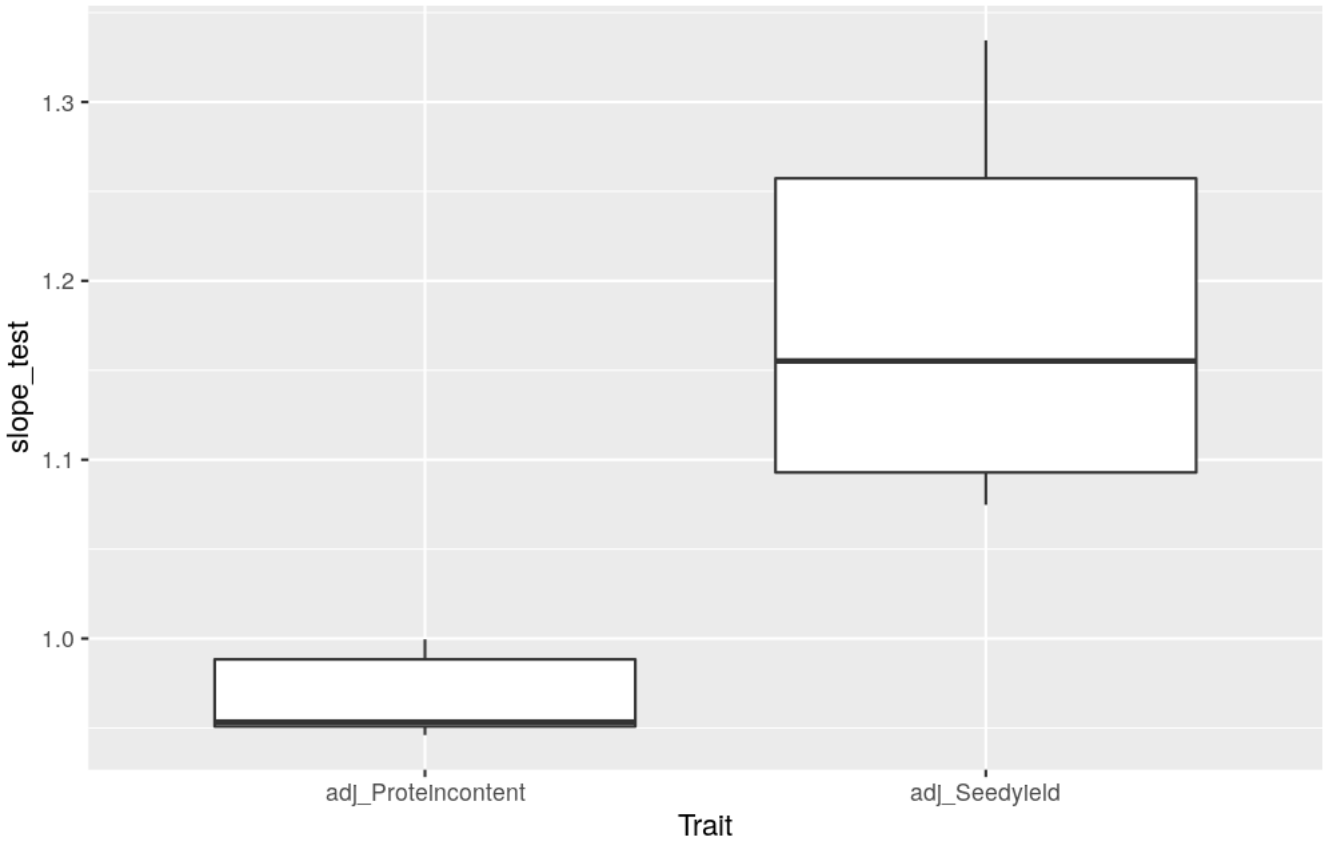
Looking at the value of the slope is also important. When the slope deviates from zero, it means that one group of individuals is less well predicted than others or that the prediction is biased in some way.

```
ggplot(Results, aes(x = Trait, y = slope_val)) + geom_boxplot()
```



Then in Test Set

```
ggplot(Results, aes(x = Trait, y = slope_test)) + geom_boxplot()
```



Now for the Training with 20% and 70% of the individuals

```
ResultsF = Results
Results = data.frame(Trait = c("adj_Proteincontent",
    "adj_Seedyield"), Model = "ST", TrainingPerc = 20,
    rep = 1:10, heritability = NA, AIC = NA, cor_val = NA,
    cor_test = NA, rank_val = NA, rank_test = NA, Acc_val = NA,
    Acc_test = NA, slope_val = NA, slope_test = NA)



for (i in 1:10) {
    ind_val = INDVAL20[[i]]  # individual as validation

    phenotp = Adj_PhenoSoy  # temporary phenotypic dataset
    # Validation and Test individuals are masked to be
    # predicted by the model with the genomic
    # information

    phenotp$adj_Proteincontent[phenotp$EUCLEGID %in%
        c(ind_test, ind_val)] = NA

    # GBLUP
    GBLUP <- remlf90(fixed = adj_Proteincontent ~ 1,
        generic = list(genetic = list(inc.H, HX2)),
        data = phenotp)
```

```
    # we are extractin GEBV (Genomic Estimated Breeding
    # Value)
    gebv = GBLUP$ranef$generic_genetic[[1]]$value
    # I put the individuals name to compare the results
    names(gebv) = Adj_PhenoSoy$EUCLEGID

    # i stock in the dataset the predicted gebv
    Adj_PhenoSoy$gebv = gebv

    # Estimation of the heritability
    Results$heritability[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = GBLUP$var["generic_genetic",
        "Estimated variances"]/sum(GBLUP$var[, "Estimated variances"])

    # Model AIC extraction
    Results$AIC[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = GBLUP$fit$AIC

    # Estimation of the Predictive ability in the
    # validation set
    Results$cor_val[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], use = "na.or.complete")

    # Estimation of the Predictive ability in the
    # Testset
    Results$cor_test[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], use = "na.or.complete")

    # Estimation of the Rank prediction in the
    # validation set
    Results$rank_val[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], use = "na.or.complete", method = "spearman")

    # Estimation of the Rank prediction in the Test set
    Results$rank_test[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], use = "na.or.complete", method = "spearman")

    # Estimation of the Accuracy in the validation set
    Results$Acc_val[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], use = "na.or.complete")/sqrt(heritabilities["Proteincontent"])

    # Estimation of the Accuracy in the Testset
    Results$Acc_test[Results$Trait == "adj_Proteincontent" &
```

```
    Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
    ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_test], use = "na.or.complete")/sqrt(heritabilities["Proteincontent"])

# Determination of the slope between phenotypes and
# gebv in the validation set
Results$slope_val[Results$Trait == "adj_Proteincontent" &
    Results$rep == i] = coef(lm(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
    ind_val] ~ Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_val]))[2]

# Determination of the slope between phenotypes and
# gebv in the Test set
Results$slope_test[Results$Trait == "adj_Proteincontent" &
    Results$rep == i] = coef(lm(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
    ind_test] ~ Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_test]))[2]

# We are doing the same for Seed yield
phenotp = Adj_PhenoSoy  # temporary phenotypic dataset
# Validation and Test individuals are masked to be
# predicted by the model with the genomic
# information

phenotp$adj_Seedyield[phenotp$EUCLEGID %in% c(ind_test,
    ind_val)] = NA

# GBLUP
GBLUP <- remlf90(fixed = adj_Seedyield ~ 1, generic = list(genetic = list(inc.H,
    HX2)), data = phenotp)

# we are extractin GEBV (Genomic Estimated Breeding
# Value)
gebv = GBLUP$ranef$generic_genetic[[1]]$value
# I put the individuals name to compare the results
names(gebv) = Adj_PhenoSoy$EUCLEGID

# i stock in the dataset the predicted gebv
Adj_PhenoSoy$gebv = gebv

# Estimation of the heritability
Results$heritability[Results$Trait == "adj_Seedyield" &
    Results$rep == i] = GBLUP$var["generic_genetic",
    "Estimated variances"]/sum(GBLUP$var[, "Estimated variances"])

# Model AIC extraction
Results$AIC[Results$Trait == "adj_Seedyield" &
    Results$rep == i] = GBLUP$fit$AIC

# Estimation of the Predictive ability in the
# validation set
Results$cor_val[Results$Trait == "adj_Seedyield" &
    Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
```

```
        ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], use = "na.or.complete")

    # Estimation of the Predictive ability in the
    # Testset
    Results$cor_test[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], use = "na.or.complete")

    # Estimation of the Rank prediction in the
    # validation set
    Results$rank_val[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], use = "na.or.complete", method = "spearman")

    # Estimation of the Rank prediction in the Test set
    Results$rank_test[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], use = "na.or.complete", method = "spearman")

    # Estimation of the Accuracy in the validation set
    Results$Acc_val[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], use = "na.or.complete")/sqrt(heritabilities["Seedyield"])

    # Estimation of the Accuracy in the Testset
    Results$Acc_test[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], use = "na.or.complete")/sqrt(heritabilities["Seedyield"])

    # Determination of the slope between phenotypes and
    # gebv in the validation set
    Results$slope_val[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = coef(lm(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_val] ~ Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val]))[2]

    # Determination of the slope between phenotypes and
    # gebv in the Test set
    Results$slope_test[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = coef(lm(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_test] ~ Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test]))[2]


}

ResultsF = rbind(ResultsF, Results)
```

```
Results = data.frame(Trait = c("adj_Proteincontent",
    "adj_Seedyield"), Model = "ST", TrainingPerc = 70,
    rep = 1:10, heritability = NA, AIC = NA, cor_val = NA,
    cor_test = NA, rank_val = NA, rank_test = NA, Acc_val = NA,
    Acc_test = NA, slope_val = NA, slope_test = NA)



for (i in 1:10) {
    ind_val = INDVAL70[[i]]  # individual as validation

    phenotp = Adj_PhenoSoy  # temporary phenotypic dataset
    # Validation and Test individuals are masked to be
    # predicted by the model with the genomic
    # information

    phenotp$adj_Proteincontent[phenotp$EUCLEGID %in%
        c(ind_test, ind_val)] = NA

    # GBLUP
    GBLUP <- remlf90(fixed = adj_Proteincontent ~ 1,
        generic = list(genetic = list(inc.H, HX2)),
        data = phenotp)

    # we are extractin GEBV (Genomic Estimated Breeding
    # Value)
    gebv = GBLUP$ranef$generic_genetic[[1]]$value
    # I put the individuals name to compare the results
    names(gebv) = Adj_PhenoSoy$EUCLEGID

    # i stock in the dataset the predicted gebv
    Adj_PhenoSoy$gebv = gebv

    # Estimation of the heritability
    Results$heritability[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = GBLUP$var["generic_genetic",
        "Estimated variances"]/sum(GBLUP$var[, "Estimated variances"])

    # Model AIC extraction
    Results$AIC[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = GBLUP$fit$AIC

    # Estimation of the Predictive ability in the
    # validation set
    Results$cor_val[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val], use = "na.or.complete")

    # Estimation of the Predictive ability in the
    # Testset
    Results$cor_test[Results$Trait == "adj_Proteincontent" &
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
```

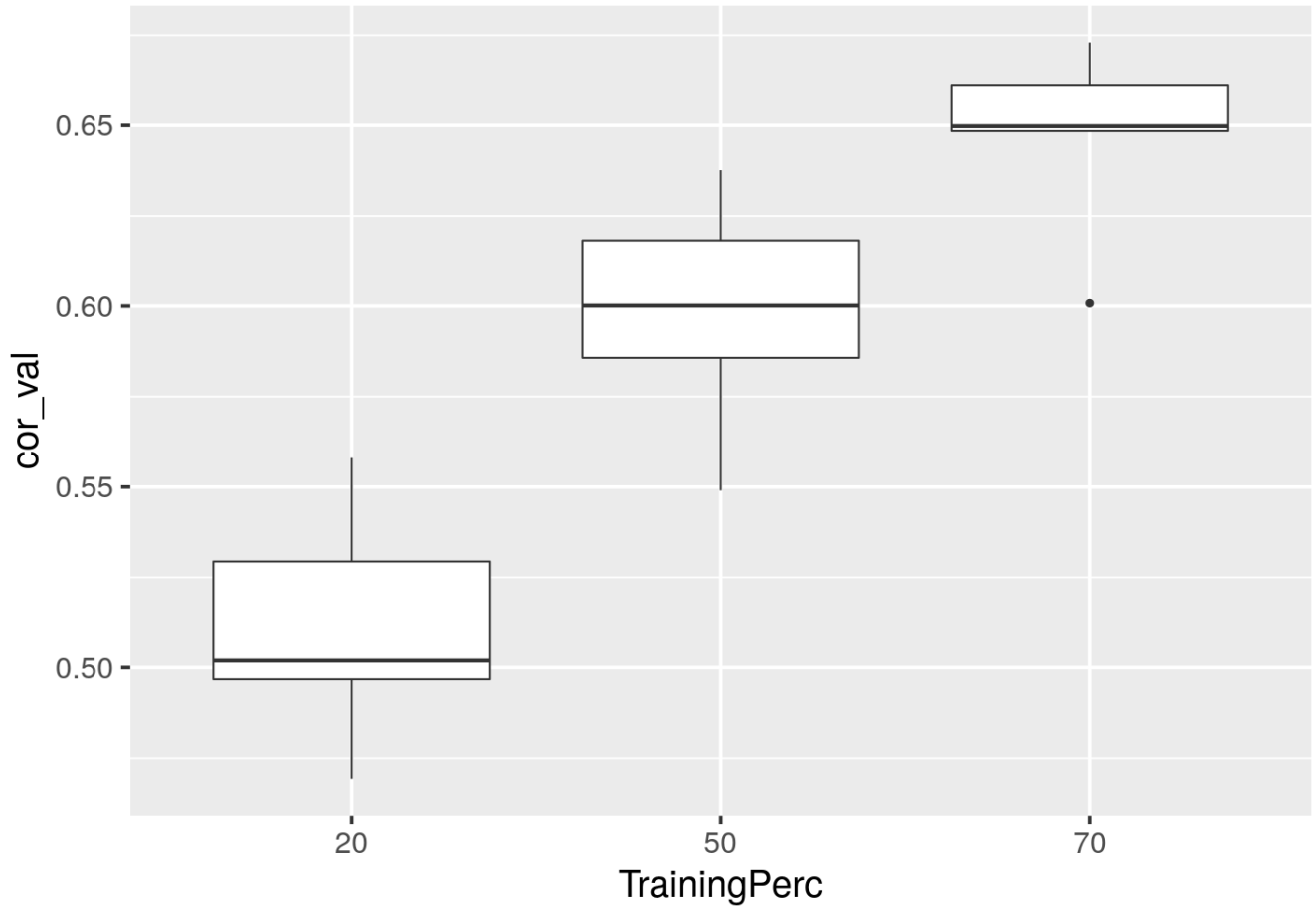
```
    ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_test], use = "na.or.complete")

# Estimation of the Rank prediction in the
# validation set
Results$rank_val[Results$Trait == "adj_Proteincontent" &
    Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
    ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_val], use = "na.or.complete", method = "spearman")

# Estimation of the Rank prediction in the Test set
Results$rank_test[Results$Trait == "adj_Proteincontent" &
    Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
    ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_test], use = "na.or.complete", method = "spearman")

# Estimation of the Accuracy in the validation set
Results$Acc_val[Results$Trait == "adj_Proteincontent" &
    Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
    ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_val], use = "na.or.complete")/sqrt(heritabilities["Proteincontent"])
# Estimation of the Accuracy in the Testset
Results$Acc_test[Results$Trait == "adj_Proteincontent" &
    Results$rep == i] = cor(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
    ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_test], use = "na.or.complete")/sqrt(heritabilities["Proteincontent"])

# Determination of the slope between phenotypes and
# gebv in the validation set
Results$slope_val[Results$Trait == "adj_Proteincontent" &
    Results$rep == i] = coef(lm(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
    ind_val] ~ Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_val]))[2]

# Determination of the slope between phenotypes and
# gebv in the Test set
Results$slope_test[Results$Trait == "adj_Proteincontent" &
    Results$rep == i] = coef(lm(Adj_PhenoSoy$adj_Proteincontent[Adj_PhenoSoy$EUCLEGID %in%
    ind_test] ~ Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_test]))[2]

# We are doing the same for Seed yield
phenotp = Adj_PhenoSoy  # temporary phenotypic dataset
# Validation and Test individuals are masked to be
# predicted by the model with the genomic
# information

phenotp$adj_Seedyield[phenotp$EUCLEGID %in% c(ind_test,
    ind_val)] = NA

# GBLUP
GBLUP <- remlf90(fixed = adj_Seedyield ~ 1, generic = list(genetic = list(inc.H,
    HX2)), data = phenotp)
```

```
# we are extractin GEBV (Genomic Estimated Breeding
# Value)
gebv = GBLUP$ranef$generic_genetic[[1]]$value
# I put the individuals name to compare the results
names(gebv) = Adj_PhenoSoy$EUCLEGID

# i stock in the dataset the predicted gebv
Adj_PhenoSoy$gebv = gebv

# Estimation of the heritability
Results$heritability[Results$Trait == "adj_Seedyield" &
    Results$rep == i] = GBLUP$var["generic_genetic",
    "Estimated variances"]/sum(GBLUP$var[, "Estimated variances"])

# Model AIC extraction
Results$AIC[Results$Trait == "adj_Seedyield" &
    Results$rep == i] = GBLUP$fit$AIC

# Estimation of the Predictive ability in the
# validation set
Results$cor_val[Results$Trait == "adj_Seedyield" &
    Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
    ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_val], use = "na.or.complete")

# Estimation of the Predictive ability in the
# Testset
Results$cor_test[Results$Trait == "adj_Seedyield" &
    Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
    ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_test], use = "na.or.complete")

# Estimation of the Rank prediction in the
# validation set
Results$rank_val[Results$Trait == "adj_Seedyield" &
    Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
    ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_val], use = "na.or.complete", method = "spearman")

# Estimation of the Rank prediction in the Test set
Results$rank_test[Results$Trait == "adj_Seedyield" &
    Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
    ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_test], use = "na.or.complete", method = "spearman")

# Estimation of the Accuracy in the validation set
Results$Acc_val[Results$Trait == "adj_Seedyield" &
    Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
    ind_val], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
    ind_val], use = "na.or.complete")/sqrt(heritabilities["Seedyield"])

# Estimation of the Accuracy in the Testset
Results$Acc_test[Results$Trait == "adj_Seedyield" &
```

```
        Results$rep == i] = cor(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test], use = "na.or.complete")/sqrt(heritabilities["Seedyield"])

    # Determination of the slope between phenotypes and
    # gebv in the validation set
    Results$slope_val[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = coef(lm(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_val] ~ Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_val]))[2]

    # Determination of the slope between phenotypes and
    # gebv in the Test set
    Results$slope_test[Results$Trait == "adj_Seedyield" &
        Results$rep == i] = coef(lm(Adj_PhenoSoy$adj_Seedyield[Adj_PhenoSoy$EUCLEGID %in%
        ind_test] ~ Adj_PhenoSoy$gebv[Adj_PhenoSoy$EUCLEGID %in%
        ind_test]))[2]


}


ResultsF = rbind(ResultsF, Results)
rm(Results)

ResultsF$TrainingPerc = as.character(ResultsF$TrainingPerc)
```

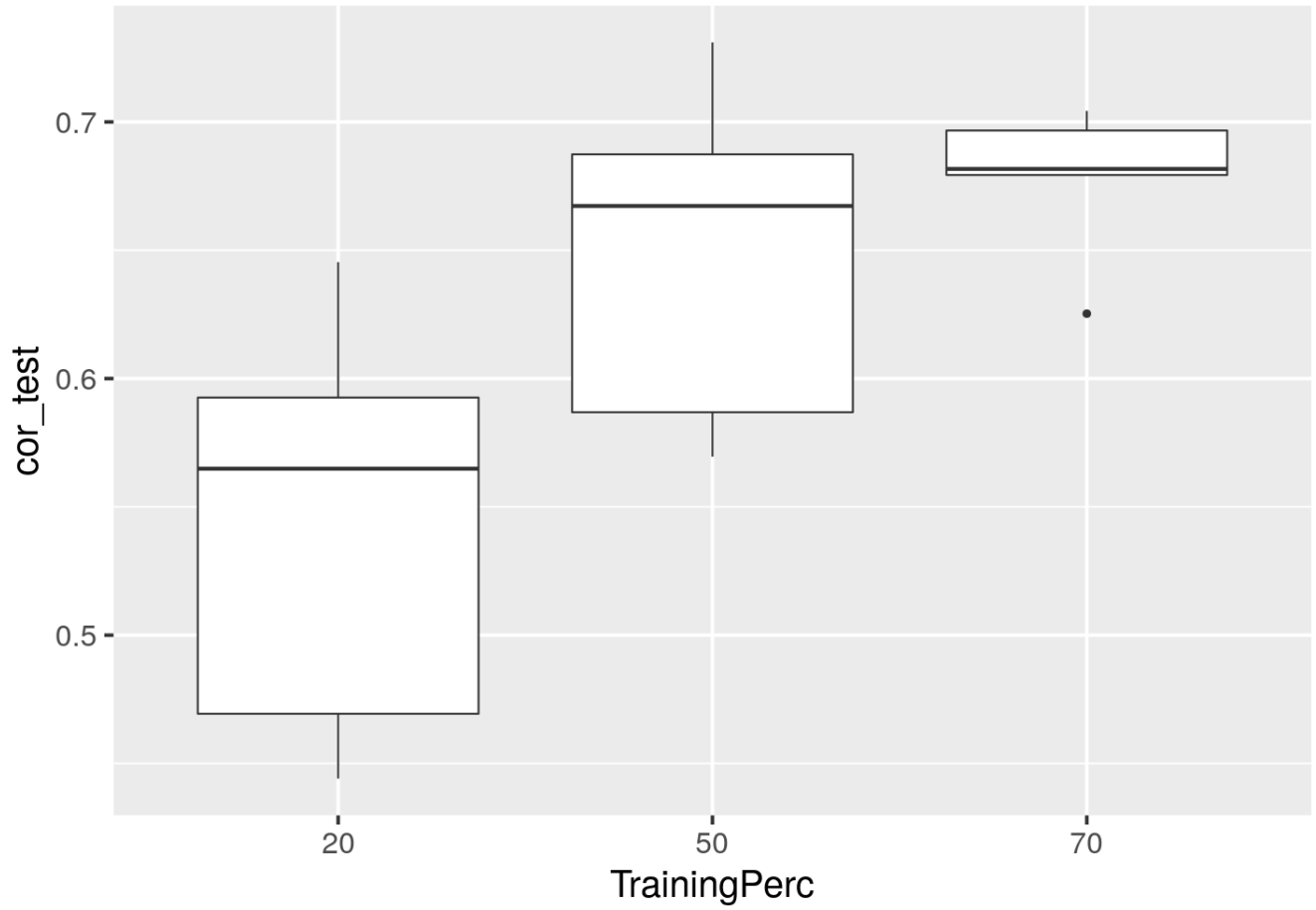
The predicting ability in validation

```
ggplot(ResultsF, aes(x = TrainingPerc, y = cor_val,
    group = TrainingPerc)) + geom_boxplot() + facet_wrap(~Trait)
```



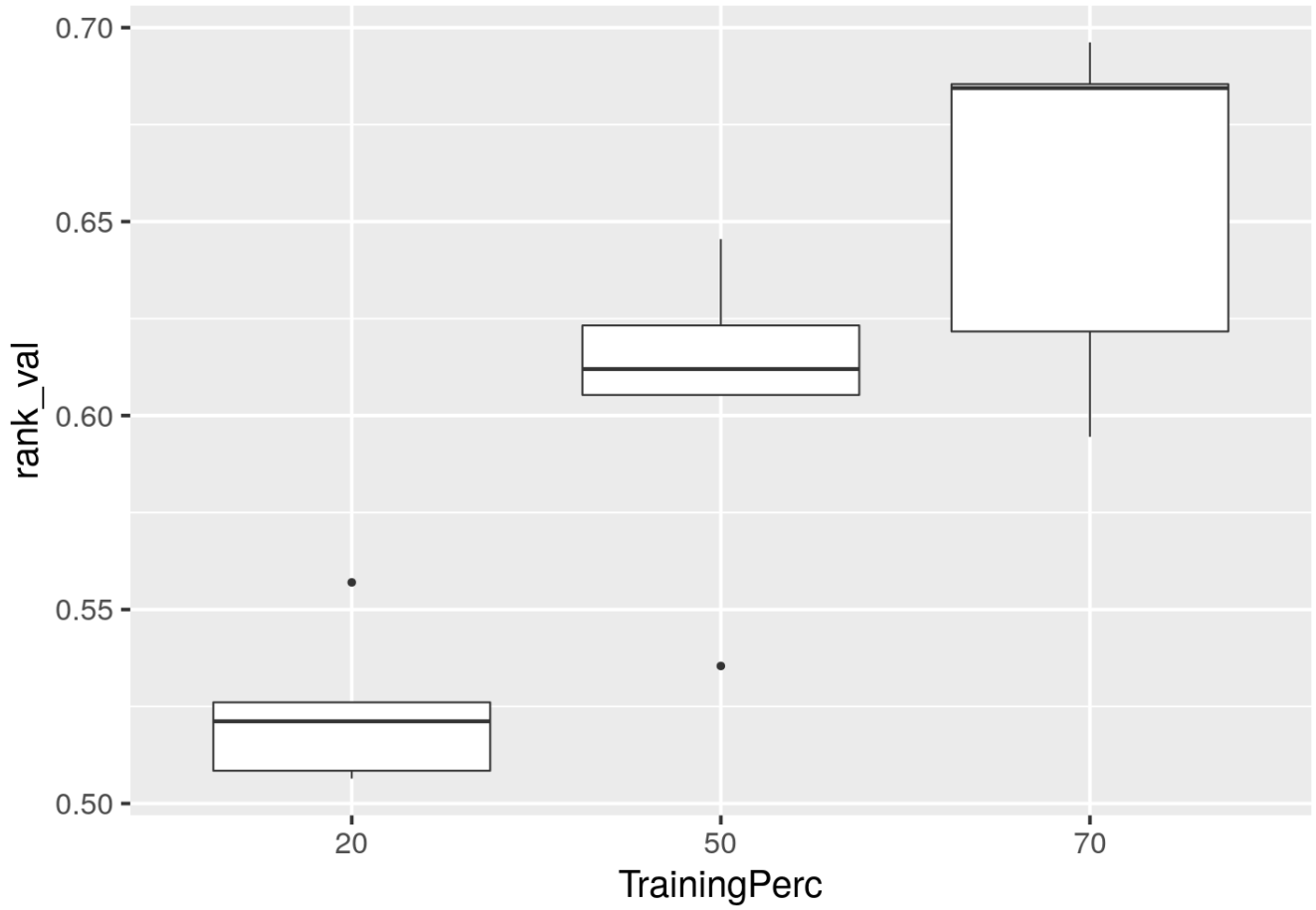
In the test Set

```
ggplot(ResultsF, aes(x = TrainingPerc, y = cor_test,
    group = TrainingPerc)) + geom_boxplot() + facet_wrap(~Trait)
```



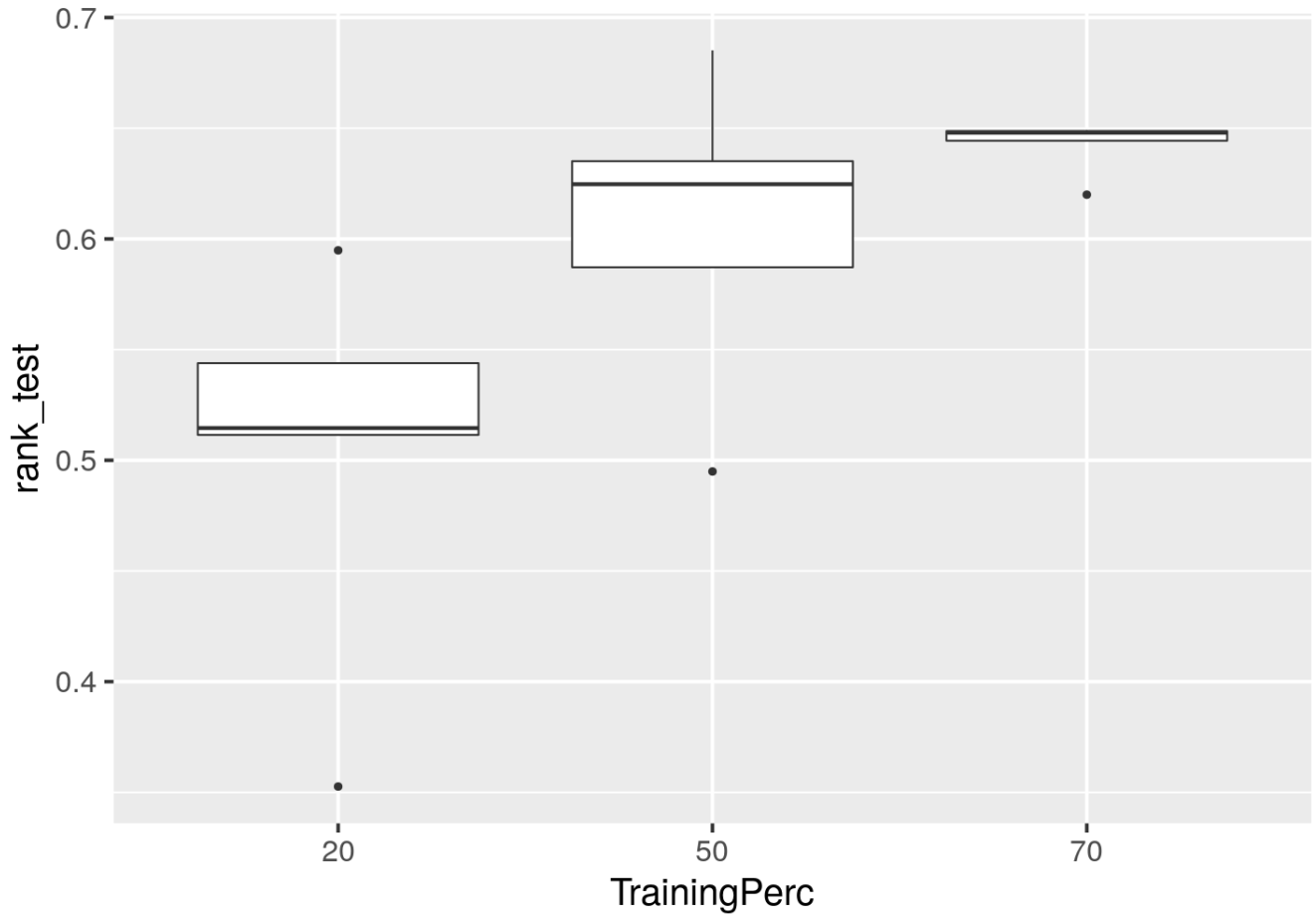
The ranking in validation

```
ggplot(ResultsF, aes(x = TrainingPerc, y = rank_val,
    group = TrainingPerc)) + geom_boxplot() + facet_wrap(~Trait)
```



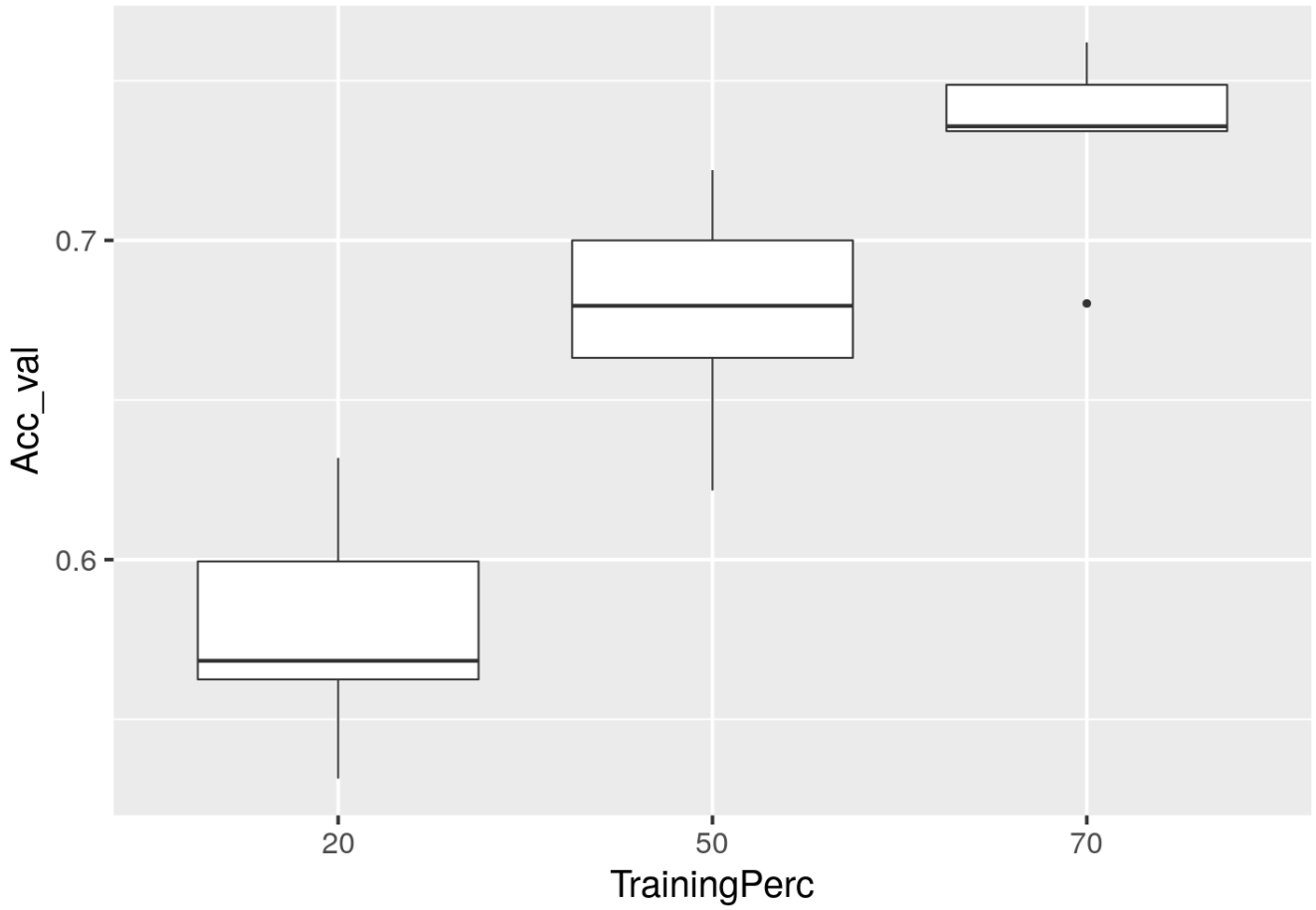
Then in Test Set

```
ggplot(ResultsF, aes(x = TrainingPerc, y = rank_test,
    group = TrainingPerc)) + geom_boxplot() + facet_wrap(~Trait)
```



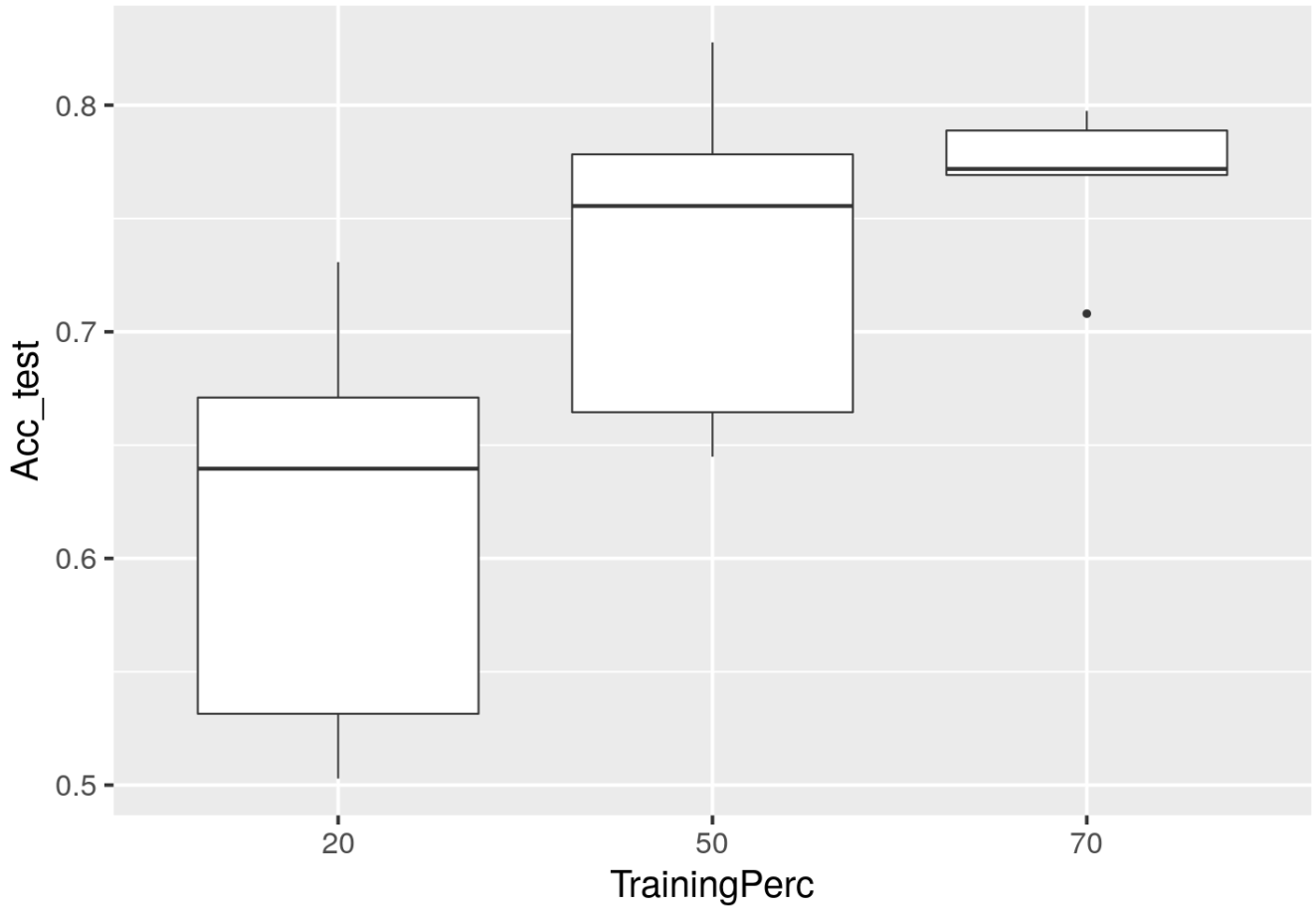
The accuracy in validation

```
ggplot(ResultsF, aes(x = TrainingPerc, y = Acc_val,
    group = TrainingPerc)) + geom_boxplot() + facet_wrap(~Trait)
```



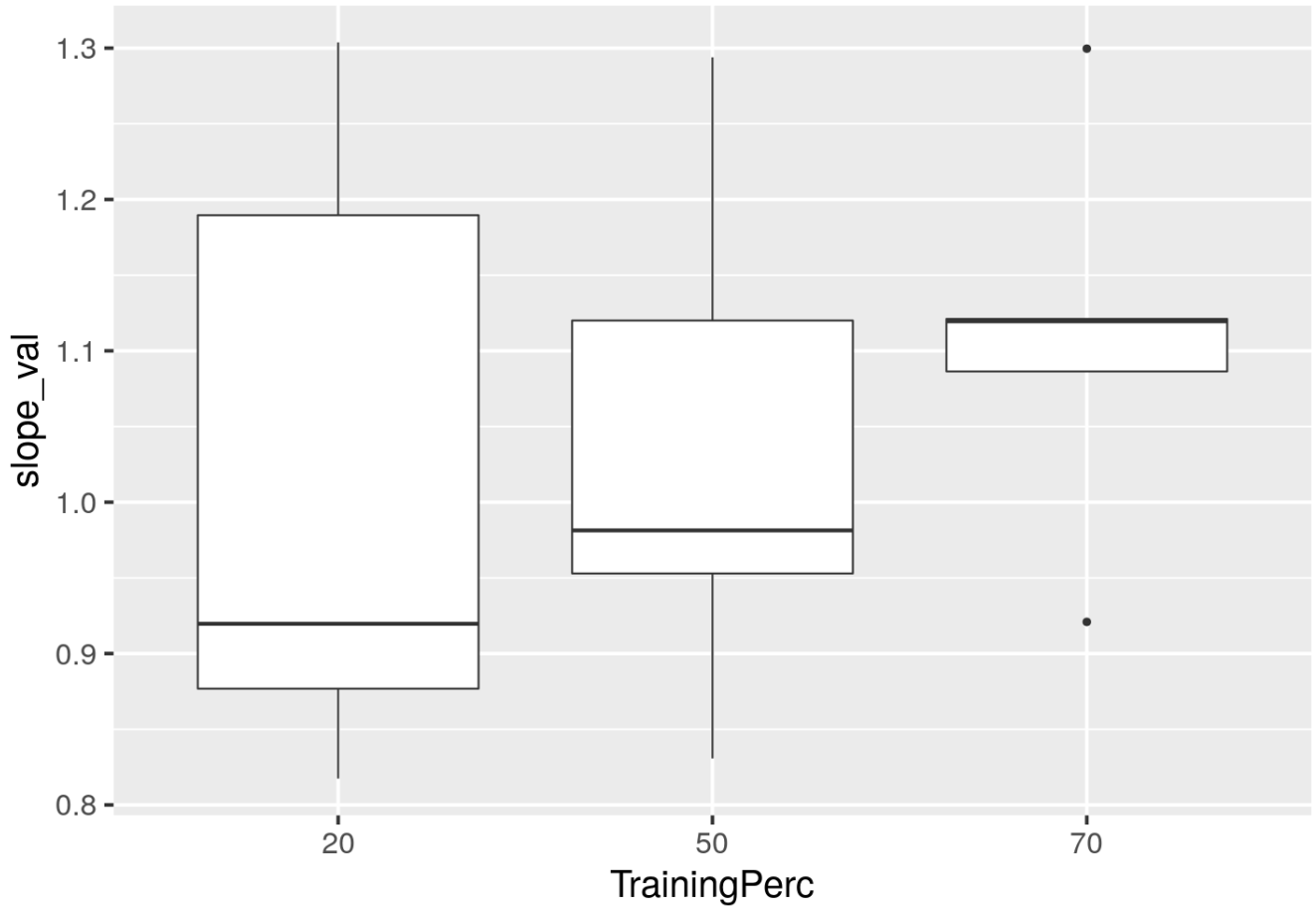
Then in Test Set

```
ggplot(ResultsF, aes(x = TrainingPerc, y = Acc_test,
    group = TrainingPerc)) + geom_boxplot() + facet_wrap(~Trait)
```



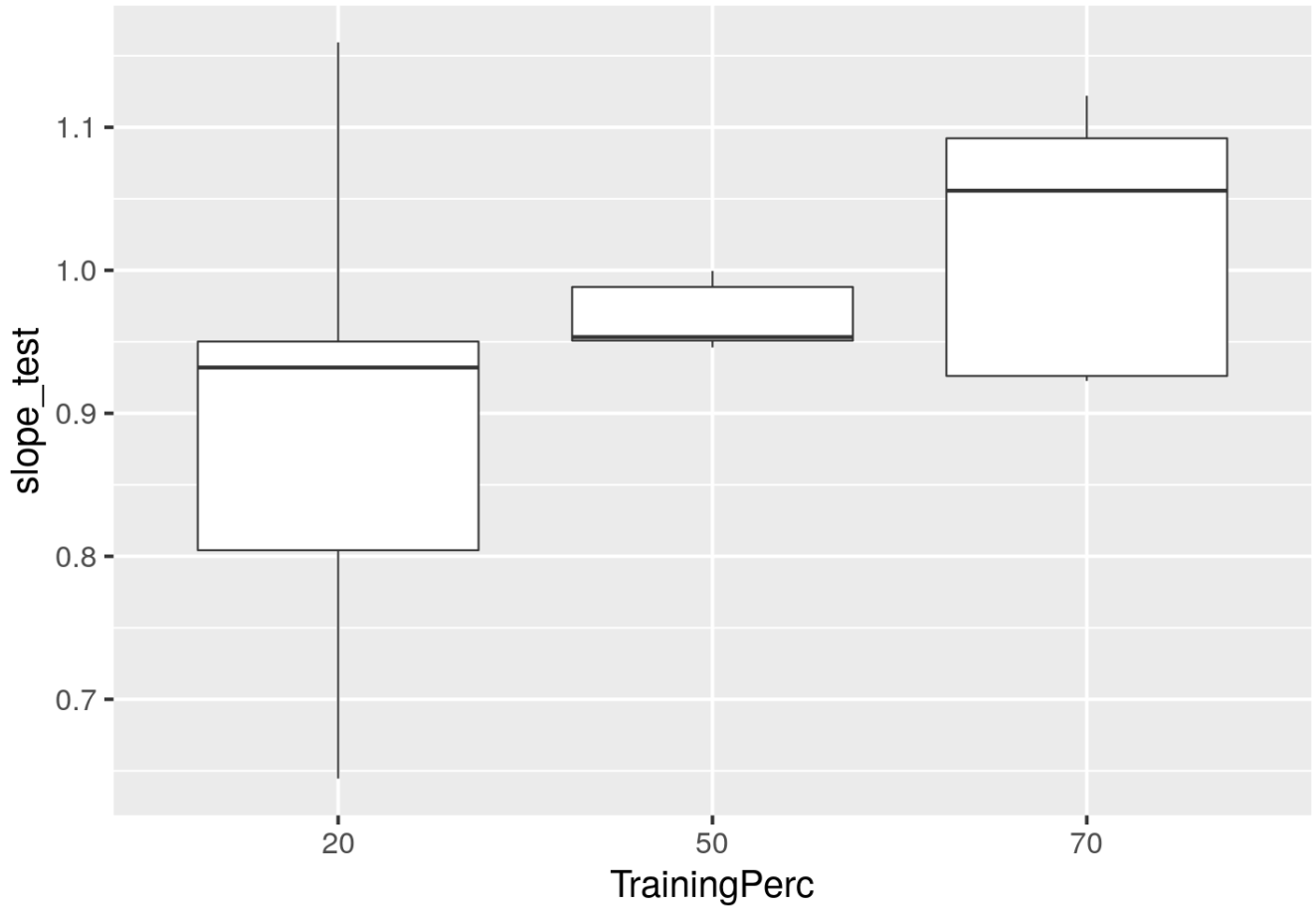
The slope in validation

```
ggplot(ResultsF, aes(x = TrainingPerc, y = slope_val,
    group = TrainingPerc)) + geom_boxplot() + facet_wrap(~Trait)
```



Then in Test Set

```
ggplot(ResultsF, aes(x = TrainingPerc, y = slope_test,
    group = TrainingPerc)) + geom_boxplot() + facet_wrap(~Trait)
```



Q-GBLUP

In the QGLUP, we will integrate the information captured by GWAS to improve the model. The detected SNPs are used as fixed effects (*_f*) *in the model or as random effects (*_r*)*. When they are fixed effects, no assertion is made about additivity or non-additivity, we try to obtain the effect of the allelic dose concerned. When they are in random effects, we are in a purely additive hypothesis on average but each individual may present variations around the average effect.

To avoid overfitting the model, the matching matrix is recalculated without the selected markers. I am only keeping the genotyped individuals in this example. I chose step 5 of the MLMM model to see the interest of adding QTLs.

```
Results1 = data.frame(Trait = c("adj_Proteincontent",
    "adj_Seedyield"), Model = "GBLUP", TrainingPerc = 50,
    rep = 1:10, heritability = NA, AIC = NA, cor_val = NA,
    cor_test = NA, rank_val = NA, rank_test = NA)

Results1b = data.frame(Trait = c("adj_Proteincontent",
    "adj_Seedyield"), Model = "QBLUP_f", TrainingPerc = 50,
    rep = 1:10, heritability = NA, AIC = NA, cor_val = NA,
    cor_test = NA, rank_val = NA, rank_test = NA)

Results1t = data.frame(Trait = c("adj_Proteincontent",
```

```
    "adj_Seedyield"), Model = "QBLUP_r", TrainingPerc = 50,
    rep = 1:10, heritability = NA, AIC = NA, cor_val = NA,
    cor_test = NA, rank_val = NA, rank_test = NA)


genotyped = rownames(M)

SNP_QTL = mygwas_proteincontent$pval_step[[5]]$cof
# Recalculation of the kinship matrix without QTL
# SNPs (VanRaden) The ! dot allows you to do the
# reverse of the command.
Z <- M[, !colnames(M) %in% SNP_QTL] - (2 * P[, !colnames(M) %in%
    SNP_QTL])

den = tcrossprod(Z)
GV = den/mean(diag(den))

# Extraction odf the SNP detected from the
# genotyping matrice
QBLUP_Data = cbind(Adj_PhenoSoy[match(genotyped, Adj_PhenoSoy$EUCLEGID),
    ], M[, SNP_QTL])

# I order G depending on the phénotypic data
GV2 = GV[match(QBLUP_Data$EUCLEGID, rownames(GV)),
    match(QBLUP_Data$EUCLEGID, colnames(GV))]
nbtot = nrow(GV2)

# I rewrite the columns with an appropriate dash
# because in formula mode it creates an error.
colnames(QBLUP_Data)[grep("AX-", colnames(QBLUP_Data))] = sub("AX-",
    "AX", colnames(QBLUP_Data)[grep("AX", colnames(QBLUP_Data))])

# I create the formulas for the model
form_r = formula(paste0(" ~ ", paste(sub("AX-", "AX",
    SNP_QTL), collapse = " + ")))
form_f = formula(paste0("adj_Proteincontent ~ 1+",
    paste(sub("AX-", "AX", SNP_QTL), collapse = " + ")))



# I resample my genotyped individuals with the same
# proportions for cross-validation.
nbtestset = nbtot * 0.2
nbval = (nbtot - nbtestset) * 0.5

ind_test = sample(QBLUP_Data$EUCLEGID, nbtestset)

inc.G <- matrix(0, nrow(QBLUP_Data), ncol = ncol(GV2))
colnames(inc.G) <- colnames(GV2)
rownames(inc.G) <- QBLUP_Data$EUCLEGID

for (i in colnames(inc.G)) {
    inc.G[which(rownames(inc.G) == i), i] <- 1
```

```
}

# Individual sampling for the cross-validation, 50%
# of the individual are training the model and the
# 50% relaing are used to validate the model
INDVALQ50 = list()
for (i in 1:10) {
    ind_val = sample(rownames(GV2)[!rownames(GV2) %in%
        ind_test], nbval)

    INDVALQ50[[i]] = ind_val

}
```

For comparison, I run a classic GBLUP with the G-matrix without the QTL SNPs. Then I use the QTLs in the model. First for protein content.

```
for(i in 1:10){
  ind_val = INDVALQ50[[i]]
  phenotp = QBLUP_Data
  phenotp[phenotp$EUCLEGID %in% c(ind_test,ind_val),
          c("adj_Proteincontent")] = NA

  GBLUP <- remlf90(
    fixed  =  adj_Proteincontent ~ 1 ,
    # random = ~  R8, # If you need random effects
    generic = list(genetic = list(inc.G,GV2)),
    data = phenotp)

  gebv = fitted(GBLUP)
  names(gebv) = QBLUP_Data$EUCLEGID

  QBLUP_Data$gebv = gebv

  Results1$heritability[Results1$Trait == "adj_Proteincontent" & Results1$rep == i]  =
    GBLUP$var["generic_genetic","Estimated variances"] / sum(GBLUP$var[,"Estimated variances"] )

  Results1$AIC[Results1$Trait == "adj_Proteincontent" & Results1$rep == i]  = GBLUP$fit$AIC

  Results1$cor_val[Results1$Trait == "adj_Proteincontent" & Results1$rep == i]  =
    cor(QBLUP_Data$adj_Proteincontent[QBLUP_Data$EUCLEGID %in% ind_val],
        QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_val],use="na.or.complete")

  Results1$cor_test[Results1$Trait == "adj_Proteincontent" & Results1$rep == i]  =
    cor(QBLUP_Data$adj_Proteincontent[QBLUP_Data$EUCLEGID %in% ind_test],
        QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_test],use="na.or.complete")

  Results1$rank_val[Results1$Trait == "adj_Proteincontent" & Results1$rep == i]  =
    cor(QBLUP_Data$adj_Proteincontent[QBLUP_Data$EUCLEGID %in% ind_val],
        QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_val],use="na.or.complete",method = "spearman")

  Results1$rank_test[Results1$Trait == "adj_Proteincontent" & Results1$rep == i]  =
    cor(QBLUP_Data$adj_Proteincontent[QBLUP_Data$EUCLEGID %in% ind_test],
        QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_test],use="na.or.complete",method = "spearman")
```

```
QGBLUP_f <- remlf90(
  fixed  =  as.formula(form_f),
  generic = list(genetic = list(inc.G,GV2)),
  data = phenotp,method = "em")

gebv = fitted(GBLUP)
names(gebv) = QBLUP_Data$EUCLEGID

QBLUP_Data$gebv = gebv

Results1b$heritability[Results1b$Trait == "adj_Proteincontent" & Results1b$rep == i]  =
  QGBLUP_f$var["generic_genetic","Estimated variances"] / sum(GBLUP$var[,"Estimated variances"] )

Results1b$AIC[Results1b$Trait == "adj_Proteincontent" & Results1b$rep == i]  = QGBLUP_f$fit$AIC

Results1b$cor_val[Results1b$Trait == "adj_Proteincontent" & Results1b$rep == i]  =
  cor(QBLUP_Data$adj_Proteincontent[QBLUP_Data$EUCLEGID %in% ind_val],
      QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_val],use="na.or.complete")

Results1b$cor_test[Results1b$Trait == "adj_Proteincontent" & Results1b$rep == i]  =
  cor(QBLUP_Data$adj_Proteincontent[QBLUP_Data$EUCLEGID %in% ind_test],
      QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_test],use="na.or.complete")

Results1b$rank_val[Results1b$Trait == "adj_Proteincontent" & Results1b$rep == i]  =
  cor(QBLUP_Data$adj_Proteincontent[QBLUP_Data$EUCLEGID %in% ind_val],
      QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_val],use="na.or.complete",method = "spearman")

Results1b$rank_test[Results1b$Trait == "adj_Proteincontent" & Results1b$rep == i]  =
  cor(QBLUP_Data$adj_Proteincontent[QBLUP_Data$EUCLEGID %in% ind_test],
      QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_test],use="na.or.complete",method = "spearman")

QGBLUP_r <- remlf90(
  fixed  =  adj_Proteincontent ~ 1,
  random =  as.formula(form_r), # If you need random effects
  generic = list(genetic = list(inc.G,GV2)),
  data = phenotp,method = "em")

gebv = fitted(QGBLUP_r)
names(gebv) = QBLUP_Data$EUCLEGID

QBLUP_Data$gebv = gebv

Results1t$heritability[Results1t$Trait == "adj_Proteincontent" & Results1t$rep == i]  =
  sum(QGBLUP_r$var[rownames(QGBLUP_r$var) != "Residual","Estimated variances"]) /
  sum(QGBLUP_r$var[,"Estimated variances"] )

Results1t$AIC[Results1t$Trait == "adj_Proteincontent" & Results1t$rep == i]  = QGBLUP_r$fit$AIC

Results1t$cor_val[Results1t$Trait == "adj_Proteincontent" & Results1t$rep == i]  =
  cor(QBLUP_Data$adj_Proteincontent[QBLUP_Data$EUCLEGID %in% ind_val],
      QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_val],use="na.or.complete")

Results1t$cor_test[Results1t$Trait == "adj_Proteincontent" & Results1t$rep == i]  =
```

```
    cor(QBLUP_Data$adj_Proteincontent[QBLUP_Data$EUCLEGID %in% ind_test],
        QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_test],use="na.or.complete")

  Results1t$rank_val[Results1t$Trait == "adj_Proteincontent" & Results1t$rep == i]  =
    cor(QBLUP_Data$adj_Proteincontent[QBLUP_Data$EUCLEGID %in% ind_val],
        QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_val],use="na.or.complete",method = "spearman")

  Results1t$rank_test[Results1t$Trait == "adj_Proteincontent" & Results1t$rep == i]  =
    cor(QBLUP_Data$adj_Proteincontent[QBLUP_Data$EUCLEGID %in% ind_test],
        QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_test],use="na.or.complete",method = "spearman")
}
```

Then for Seed yield.

```
SNP_QTL = mygwas_Seedyield$pval_step[[5]]$cof
# Recalculation of the kinship matrix without QTL SNPs (VanRaden)
# The ! dot allows you to do the reverse of the command.
Z <-  M[,!colnames(M) %in% SNP_QTL]  - (2*P[,!colnames(M) %in% SNP_QTL])

den = tcrossprod(Z)
GV = den/mean(diag(den))

# Extraction odf the SNP detected from the genotyping matrice
QBLUP_Data = cbind(Adj_PhenoSoy[match(genotyped,Adj_PhenoSoy$EUCLEGID),], M[,SNP_QTL])

# I order G depending on the phénotypic data
GV2 = GV[match(QBLUP_Data$EUCLEGID,rownames(GV)),match(QBLUP_Data$EUCLEGID,colnames(GV))]
nbtot = nrow(GV2)

# I rewrite the columns with an appropriate dash because in formula mode it creates an error.
colnames(QBLUP_Data)[grep("AX-",colnames(QBLUP_Data))] =
  sub("AX-","AX",colnames(QBLUP_Data)[grep("AX",colnames(QBLUP_Data))])

# I create the formulas for the model
form_r = formula(paste0(" ~ ",paste(sub("AX-","AX",SNP_QTL),collapse = " + ") ))
form_f = formula(paste0("adj_Seedyield ~ 1+",paste(sub("AX-","AX",SNP_QTL),collapse = " + ") ))

for(i in 1:10){
  ind_val =  INDVALQ50[[i]]

  phenotp = QBLUP_Data
  phenotp[phenotp$EUCLEGID %in% c(ind_test,ind_val),
          c("adj_Seedyield")] = NA

  GBLUP <- remlf90(
    fixed  =  adj_Seedyield ~ 1 ,
    # random = ~  R8, # If you need random effects
    generic = list(genetic = list(inc.G,GV2)),
    data = phenotp)

  gebv = fitted(GBLUP)
  names(gebv) = QBLUP_Data$EUCLEGID

  QBLUP_Data$gebv = gebv
```

```
Results1$heritability[Results1$Trait == "adj_Seedyield" & Results1$rep == i]  =
  GBLUP$var["generic_genetic","Estimated variances"] / sum(GBLUP$var[,"Estimated variances"] )

Results1$AIC[Results1$Trait == "adj_Seedyield" & Results1$rep == i]  = GBLUP$fit$AIC

Results1$cor_val[Results1$Trait == "adj_Seedyield" & Results1$rep == i]  =
  cor(QBLUP_Data$adj_Seedyield[QBLUP_Data$EUCLEGID %in% ind_val],
      QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_val],use="na.or.complete")

Results1$cor_test[Results1$Trait == "adj_Seedyield" & Results1$rep == i]  =
  cor(QBLUP_Data$adj_Seedyield[QBLUP_Data$EUCLEGID %in% ind_test],
      QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_test],use="na.or.complete")

Results1$rank_val[Results1$Trait == "adj_Seedyield" & Results1$rep == i]  =
  cor(QBLUP_Data$adj_Seedyield[QBLUP_Data$EUCLEGID %in% ind_val],
      QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_val],use="na.or.complete",method = "spearman")

Results1$rank_test[Results1$Trait == "adj_Seedyield" & Results1$rep == i]  =
  cor(QBLUP_Data$adj_Seedyield[QBLUP_Data$EUCLEGID %in% ind_test],
      QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_test],use="na.or.complete",method = "spearman")

QGBLUP_f <- remlf90(
  fixed  =  as.formula(form_f),
  generic = list(genetic = list(inc.G,GV2)),
  data = phenotp,method = "em")

gebv = fitted(GBLUP)
names(gebv) = QBLUP_Data$EUCLEGID

QBLUP_Data$gebv = gebv

Results1b$heritability[Results1b$Trait == "adj_Seedyield" & Results1b$rep == i]  =
  QGBLUP_f$var["generic_genetic","Estimated variances"] / sum(GBLUP$var[,"Estimated variances"] )

Results1b$AIC[Results1b$Trait == "adj_Seedyield" & Results1b$rep == i]  = QGBLUP_f$fit$AIC

Results1b$cor_val[Results1b$Trait == "adj_Seedyield" & Results1b$rep == i]  =
  cor(QBLUP_Data$adj_Seedyield[QBLUP_Data$EUCLEGID %in% ind_val],
      QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_val],use="na.or.complete")

Results1b$cor_test[Results1b$Trait == "adj_Seedyield" & Results1b$rep == i]  =
  cor(QBLUP_Data$adj_Seedyield[QBLUP_Data$EUCLEGID %in% ind_test],
      QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_test],use="na.or.complete")

Results1b$rank_val[Results1b$Trait == "adj_Seedyield" & Results1b$rep == i]  =
  cor(QBLUP_Data$adj_Seedyield[QBLUP_Data$EUCLEGID %in% ind_val],
      QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_val],use="na.or.complete",method = "spearman")

Results1b$rank_test[Results1b$Trait == "adj_Seedyield" & Results1b$rep == i]  =
  cor(QBLUP_Data$adj_Seedyield[QBLUP_Data$EUCLEGID %in% ind_test],
      QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_test],use="na.or.complete",method = "spearman")

QGBLUP_r <- remlf90(
```

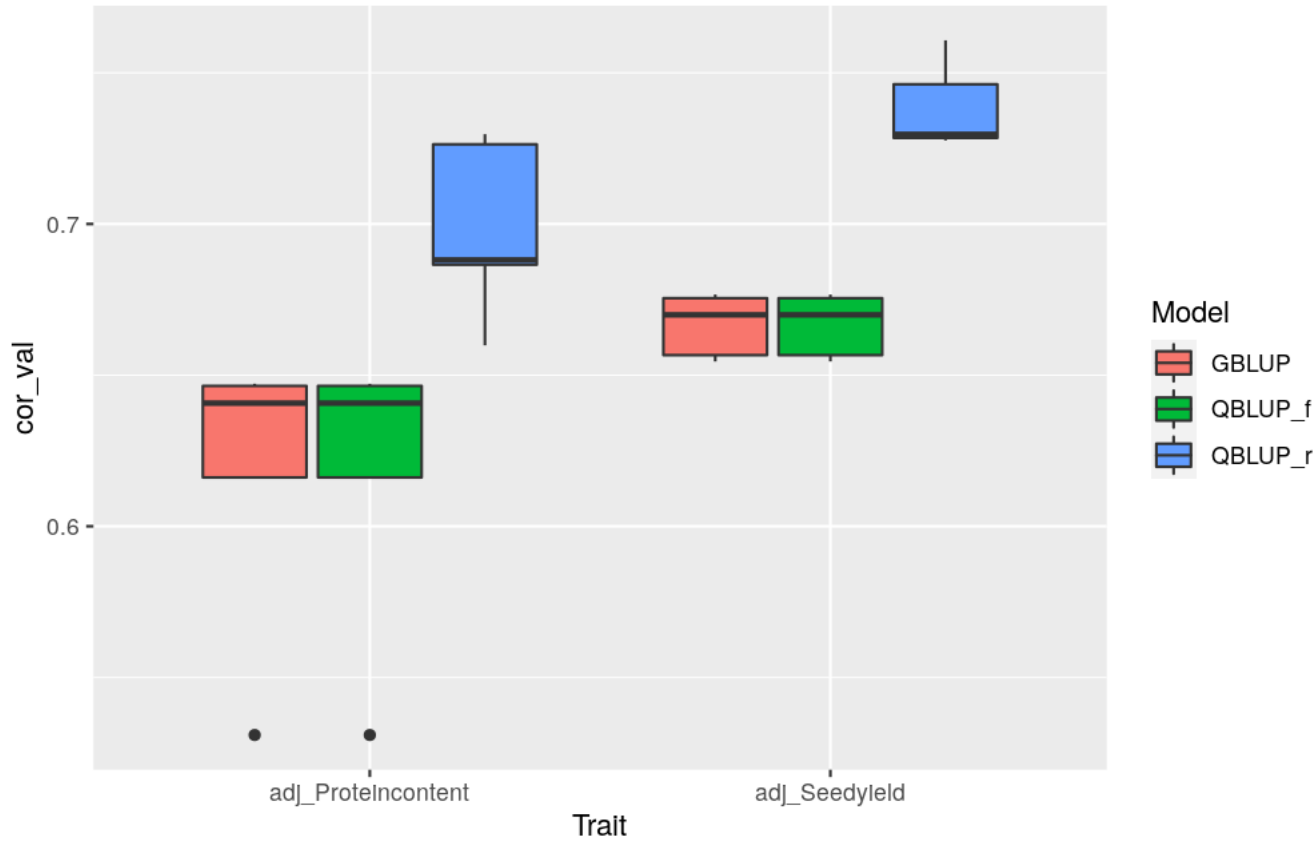
```
    fixed  =  adj_Seedyield ~ 1,
    random =  as.formula(form_r), # If you need random effects
    generic = list(genetic = list(inc.G,GV2)),
    data = phenotp,method = "em")

  gebv = fitted(QGBLUP_r)
  names(gebv) = QBLUP_Data$EUCLEGID

  QBLUP_Data$gebv = gebv

  Results1t$heritability[Results1t$Trait == "adj_Seedyield" & Results1t$rep == i]  =
    sum(QGBLUP_r$var[rownames(QGBLUP_r$var) != "Residual","Estimated variances"]) /
    sum(QGBLUP_r$var[,"Estimated variances"] )


  Results1t$AIC[Results1t$Trait == "adj_Seedyield" & Results1t$rep == i]  = QGBLUP_r$fit$AIC

  Results1t$cor_val[Results1t$Trait == "adj_Seedyield" & Results1t$rep == i]  =
    cor(QBLUP_Data$adj_Seedyield[QBLUP_Data$EUCLEGID %in% ind_val],
        QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_val],use="na.or.complete")

  Results1t$cor_test[Results1t$Trait == "adj_Seedyield" & Results1t$rep == i]  =
    cor(QBLUP_Data$adj_Seedyield[QBLUP_Data$EUCLEGID %in% ind_test],
        QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_test],use="na.or.complete")

  Results1t$rank_val[Results1t$Trait == "adj_Seedyield" & Results1t$rep == i]  =
    cor(QBLUP_Data$adj_Seedyield[QBLUP_Data$EUCLEGID %in% ind_val],
        QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_val],use="na.or.complete",method = "spearman")

  Results1t$rank_test[Results1t$Trait == "adj_Seedyield" & Results1t$rep == i]  =
    cor(QBLUP_Data$adj_Seedyield[QBLUP_Data$EUCLEGID %in% ind_test],
        QBLUP_Data$gebv[QBLUP_Data$EUCLEGID %in% ind_test],use="na.or.complete",method = "spearman")
}
```

I concatenate the data in a single object

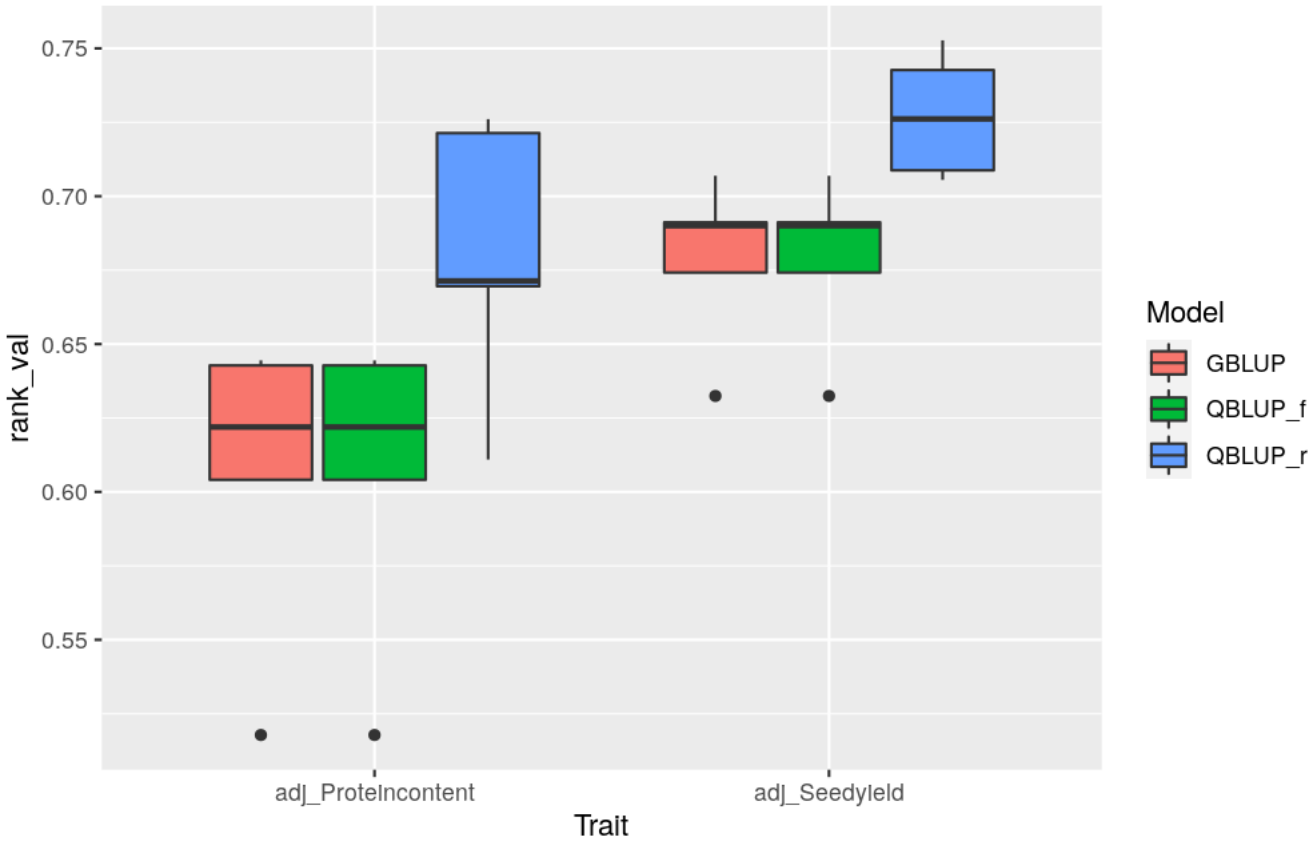
```
ResultsQ = rbind(Results1, Results1b, Results1t)
```

Let's look first for validation

```
ggplot(ResultsQ, aes(x = Trait, y = cor_val, fill = Model)) +
    geom_boxplot()
```

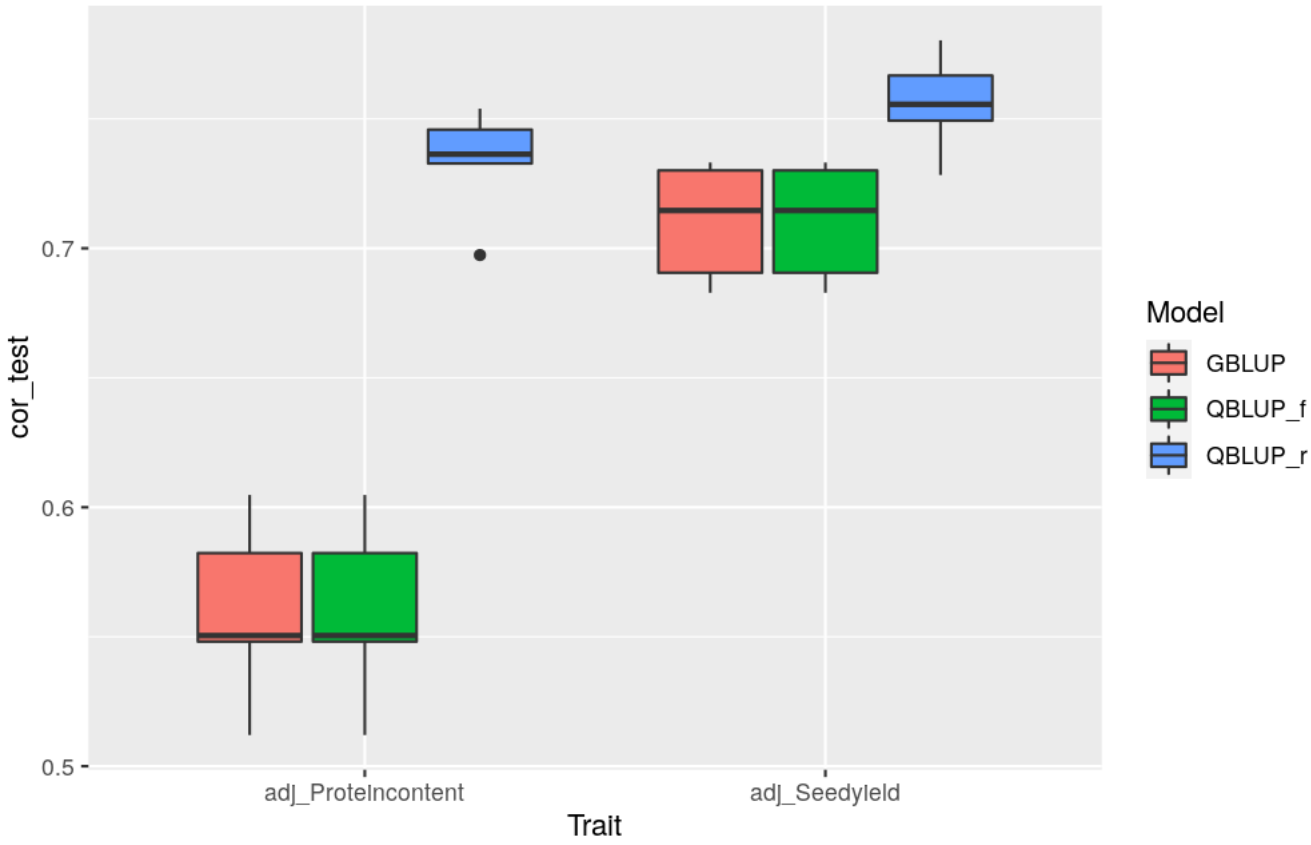



```
ggplot(ResultsQ, aes(x = Trait, y = rank_val, fill = Model)) +
    geom_boxplot()
```

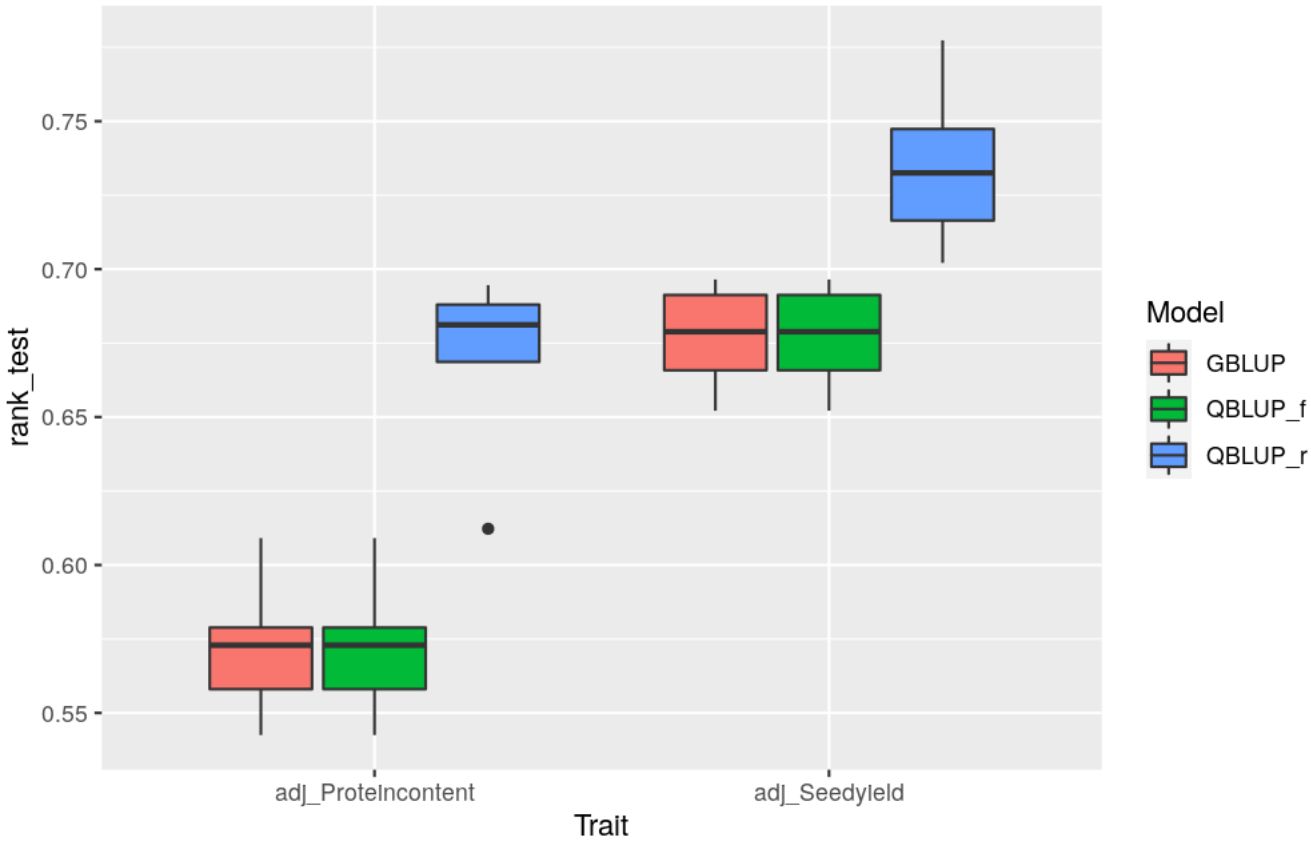


And now for the Test Set

```
ggplot(ResultsQ, aes(x = Trait, y = cor_test, fill = Model)) +
    geom_boxplot()
```




```
ggplot(ResultsQ, aes(x = Trait, y = rank_test, fill = Model)) +
    geom_boxplot()
```




```
t2 = Sys.time()
t2 - t1
```

```
## Time difference of 56.05172 mins
```

Here the results show that the addition of QTL SNPs as random effects allows a better prediction, whatever the trait.

Conclusion

To conclude this workshop, I would like to recall the important messages of this one. First of all, take the time to look at and clean up the phenotyping data as well as the genotyping data. Having clean data is necessary in order to have robust results and not make mistakes that will prove costly later on. I hope I helped you to understand what is going on with the functions of the all-inclusive packages so that you are comfortable with the choices that are being made. This workshop was made for the example, some parts of the scripts presented need optimization. I sometimes made arbitrary choices for the example, but which is justified elsewhere.