GWAS JRL 2021

J. David sur base d un script de Vincent Segura

17/03/2021

# Contexte

knitr::opts\_chunk$set(echo = TRUE, cache.lazy = FALSE)  
  
#library("rstudioapi") # mettre la directory en place  
#setwd(,n sx(getActiveDocumentContext()$path)) # Set working directory to source file location  
#getwd()

rm(list = ls())  
  
library(anyLib)  
anyLib(c("data.table", "apercu", "mlmm.gwas", "corpcor","plyr"))

## Loading required package: data.table

## Loading required package: apercu

## Loading required package: mlmm.gwas

## Loading required package: corpcor

## Loading required package: plyr

## data.table apercu mlmm.gwas corpcor plyr   
## TRUE TRUE TRUE TRUE TRUE

library(stringr)

# Introduction à la génétique d’association

Il n’est pas question ici de faire un cours sur la génétique d’association mais d’en expliquer rapidement le principe et d’expliquer le script par étape. La première étape est d’avoir obtenu sur les mêmes lignées des données phénotypiques et des données génotypiques sur un grand nombre de marqueurs, supposés représenter tous les morceaux de chromosomes dans lesquels gènes et marqueurs sont liés par le déséquilibre de liaison.

Pour simplifier, la valeur phénotypique est unique par lignée, elle a été obtenue dans une phase précédente qui a consisté à prendre en considération le plan d’expérience, les corrections spatiales éventuelles. Ce sont en général des BLUPS ou des BLUES.

Les valeurs des phénotypes sont modélisés de la manière suivante

où Y est un vecteur d’une colonne et de lignes qui porte les valeurs des phénotypes,  
X est le vecteur qui donne le nombre de copies dans chaque lignée de l’allele de référence, mettons A, à un locus où on observe seulement deux allèles comme c’est le cas des marqueurs SNP. Donc X peut prendre comme valeur 2 si la lignée est AA à ce locus, 1 si elle est hétérozygote Aa et 0 si elle est aa.  
 est la pente de regression qui modifie le phénotype en fonction du nombre d’alléles A donné par X. L’effet génétique qui est modélisé est donc complètement additif. Il est déclaré comme effet fixe. C’est à dire que plus la lignée a de copies de A, plus son phénotype est modifié en proportion, en suivant la pente . Donc vous aurez compris que c’est la valeur de qui détermine si il y a une association entre le marqueur et le phénotype. Il faut donc faire autant d’analyses qu’il y a de marqueurs et tester la pvalue de la nullité de la pente de chacun des marqueurs. L’hypothèse nulle est .

Z est une matrice de design qui attribue chaque valeur phénotypique à un effet multigénique pour chaque lignée. Cet effet, correspond à la somme de tous les effets additifs des autres gènes polymorphes dans le génome et qui agissent sur le caractère. Ici elle est simple puisqu’il n’y a qu’une seule valeur Y pour chaque .  
 est un vecteur qui donne les valeurs additives polygéniques pour chaque lignée. Comme les lignées ne sont pas indépendantes, elles peuvent être apparentées et donc leur ne sont pas indépendantes les unes des autres. Il faut donc prendre en compte le fait que posséder des allèles en commun va les faire se ressembler (leur valeur de Y sont corrélées). Donc est déclaré comme un effet aléatoire, suivant une loi normale de variance où est la variance additive polygénique du caractère et une matrice d’apparentemnt entre les lignées. K sera calculée avec les marqueurs.  
 est la résiduelle, aléatoire, de variance

Le script va donc effectuer les analyses pour tous les marqueurs et va même aller plus loin. Il va faire une analyse stepwise, en ajoutant les marqueurs qu’il a trouvé significatifs un par un, de manière successive (le plus significatif en premier) pour tester si l’ajout d’un marqueur permet de diminuer .

Comme il y a eu de très nombreuses analyses (une par marqueur), il faut être prudent sur la valeur seuil des P-values pour décider si la pente d’un marqueur ne suit pas l’hypothèse nulle et que donc l’effet du marqueur est significatif et que nous sommes bien dans une zone QTL (quantitative trait locus). Rappelez-vous qu’avec un risque de 5% de première espèce de rejeter alors qu’elle est vraie fera qu’avec 100 000 marqueurs réellement indépendants, il y aura associations déclarées significatives alors qu’il n’y en a aucune.  
Il faut donc prendre un risque plus faible pour controler pour les tests multiples. Si tous les marqueurs étaient indépedants, une solution serait d’appliquer une correction dite de Bonferroni, en divisant 5% par le nombre de marqueurs. Dans notre cas, le seuil devrait être de . Mais comme les marqueurs sont liés par haplotypes dans le génome, sans rentrer dans les détails, je vous propose de prendre comme seuil de sécurité. Il correspond à une estimation du nombre de clusters indépendants dans la population EPO.

# Passons à la pratique

## les données sur le génotype

### Positions des SNP sur les chromosomes

Les marqueurs sont positionnés sur le génome du blé dur. Il faut donc charger le fichier des positions qui ont été obtenues de manière indirecte en blastant les séquences des marqueurs sur le génome entier. Il peut y avoir plusieurs endroits candidats. Donc attention à ne pas l’oublier.

# physical positions of SNP on the Zavitan WEW2 version  
load("../data/GENOTYPES/BREEDWHEAT\_on\_durum\_physic\_WEW2.Rdata")  
BLAST[1,]

## Marker\_ID Chr Mstart Mend Chrlen start end v8 long simil  
## 1 AX-89309359 chr7A 1 71 71 742440137 742440067 747227478 71 97.18  
## evalue  
## 1 2e-25

Il y a 242382 SNP pour lesquels on a une position possible.

dim(BLAST)

## [1] 242382 11

Ces SNPs sont caractérisés par de nombreuses propriétés qui seront utilisées plus tard.

# toutes les infos dispo sur les snp  
load("../data/GENOTYPES/caract\_SNP\_ALL\_PHYS.Rdata")

## Matrice de génotypage

Les lignées ont été génotypées par une puce à ADN sur laquelle on peut lire 420 000 SNP. Les données que nous allons utiliser font partie d’un projet plus large, il faut récupérer juste les données utiles.

# breed wheat genotypes  
file="../data/GENOTYPES/SG\_EPO\_complet.Rdata"  
load(file)  
  
dim(SG)

## [1] 476 168725

Il y a donc 476 lignées sur 168 725 SNP.

On ne garde que ceux pour lesquels on a une position physique. Vous pourrez vérifier ensuite si vous auriez des associations avec des SNP dont vous n’avez pas idée de l’endroit où ils se trouvent. Il faudra changer le code qui suit dans ce cas.

# optional Keep sNP only with an assumed physical position  
liste<-which(colnames(SG) %in% BLAST[,1])  
length(liste)

## [1] 112819

ap(liste)

## [1] 1 4 7 8 101

SG<-SG[,liste]  
dim(SG)

## [1] 476 112819

# Phenotypes

## Les données

Ici il faudra placer vos propres données de BLUPs ou BLUE.

#file<-"../data/PHENOTYPES/BLUP\_example.csv"  
file<-"../data/PHENOTYPES/BlueBlupS2.csv"  
  
myY <-read.table(file, head = TRUE, sep=";", dec=",")  
  
names(myY)[1]<-"Taxa"  
  
dim(myY)

## [1] 184 3

myY[1:10, 1]

## [1] C495 ELAX\_100 ELAX\_101 ELAX\_104 ELAX\_114 ELAX\_117 ELAX\_118 ELAX\_120  
## [9] ELAX\_122 ELAX\_124  
## 184 Levels: C495 ELAX\_100 ELAX\_101 ELAX\_104 ELAX\_114 ELAX\_117 ... MariaB6R

# a cause d'4line qui confond les 4 avec des A...  
myY[,1]<-str\_replace(as.character(myY[,1]), "A", "4")

Attention à la manière dont les codes sont donnés. Il faut que ce soit codé comme dans le fichier de génotypes.

# Keep Genotypes only if phenotyped

SG<-SG[which(rownames(SG) %in% myY[,1]),]  
ap(SG)

## AX-89309359 AX-89309363 AX-89309376 AX-89309378 AX-89309726  
## EL4X\_240 "2" "2" "2" "2" "2"   
## EL4X\_356 "2" "2" "0" "2" "2"   
## EL4X\_476 "2" "2" "2" "2" "0"   
## EL4X\_241 "2" "2" "2" "2" "2"   
## EL4X\_16 "2" "2" "0" "2" "0"

dim(SG)

## [1] 181 112819

Ici il reste 178 génotypes. Et attention aussi à ne pas garder de données phénotypiques pour lesquelles nous n’avons pas de données génotypiques.

# attention il y a des génotypes qui ne sont pas dans SG  
myY<-myY[which(myY[,1] %in% rownames(SG) ),]  
dim(myY)

## [1] 181 3

dim(SG)

## [1] 181 112819

Les deux nombres ici doivent être égaux.

## Prétraitement des données

genot<-SG  
  
# il faut que genot soit une matrice  
class(genot)

## [1] "matrix"

dim(genot)

## [1] 181 112819

ap(genot)

## AX-89309359 AX-89309363 AX-89309376 AX-89309378 AX-89309726  
## EL4X\_240 "2" "2" "2" "2" "2"   
## EL4X\_356 "2" "2" "0" "2" "2"   
## EL4X\_476 "2" "2" "2" "2" "0"   
## EL4X\_241 "2" "2" "2" "2" "2"   
## EL4X\_16 "2" "2" "0" "2" "0"

### Imputation of Missing data

* simple binomial Imputation

noms<-rownames(genot)  
genot<-apply(genot,2,as.numeric)  
rownames(genot)<-noms  
  
genot.imp <- apply(genot, 2, function(x){  
 freq <- table(x)  
 x[is.na(x)] <- as.integer(names(which.max(freq)))  
 return(x)  
})

### Fréquence allélique minimale

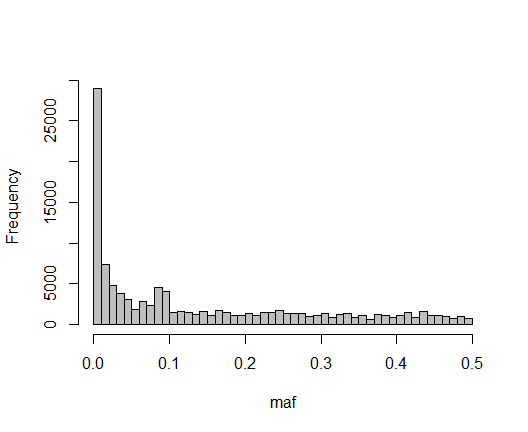
Il n’est pas raisonnable de faire une analyse si les fréquences alléliques sont trop déséquilibrées, par exemple un allèle très fréquent et l’autre très rare. Il y aurait un trop grand déséquilibre entre les effectifs des AA et des aa et la pente serait mal estimée, il pourrait y avoir des associations créées juste par quelques indididus au hasard. Donc on ne va garder que les marqueurs pour lesquels la fréquence de l’allèle minoritaire est plus grande qu’un certain seuil.

Le calcul des fréquences est facile vus comment sont codés les génotypes.

p <- colMeans(genot.imp) / 2  
q <- 1 - p

La répresentation du spectre de fréquences alléliques

maf <- apply(cbind(p, q), 1, min)  
hist(maf, col = "grey", main = "", breaks = 50, xlim = c(0, 0.5))



On voit sur cette figure que les alléles rares sont fréquents. Ici on filtre de manière à ne garder que des MAF supérieures à 5%.

sum(maf < 0.5)

## [1] 112764

genot.ok <- genot.imp[, maf >= 0.05]  
dim(genot.ok)

## [1] 181 64713

Il ne reste plus que 65405 marqueurs dans cet exemple.

### Physical map

Une opération importante est de créer le fichier de la carte physique.

map<-BLAST[which(BLAST[,1] %in% colnames(genot.ok)),c(1,2,6)]  
names(map)<-c("SNP","Chr","Pos")  
head(map)

## SNP Chr Pos  
## 8 AX-89309369 chr4B 631948705  
## 10 AX-89309371 chr1B 43228153  
## 18 AX-89309376 chr7A 605815984  
## 27 AX-89309722 chr5A 578939825  
## 32 AX-89309726 chr2A 7484084  
## 124 AX-89309734 chr3B 762124628

on ne garde que les SNP sélectionnés et on renomme les chromosomes de 1 à 14.

map <- map[map$SNP %in% colnames(genot.ok), ]  
  
map[,2] <-mapvalues(map[,2], from=c( "chrom",   
 "chr1A", "chr1B", "chr2A", "chr2B",  
 "chr3A", "chr3B", "chr4A", "chr4B",  
 "chr5A", "chr5B", "chr6A", "chr6B",  
 "chr7A", "chr7B")  
 , to=c("chrom", 1:14))

## The following `from` values were not present in `x`: chrom

map[,2]<-as.numeric(map[,2])  
  
dim(map)

## [1] 64713 3

head(map)

## SNP Chr Pos  
## 8 AX-89309369 8 631948705  
## 10 AX-89309371 2 43228153  
## 18 AX-89309376 13 605815984  
## 27 AX-89309722 9 578939825  
## 32 AX-89309726 3 7484084  
## 124 AX-89309734 6 762124628

tail(map)

## SNP Chr Pos  
## 1180551 AX-89778312 9 707489011  
## 1180554 AX-89778316 1 16997776  
## 1180555 AX-89778318 3 355542057  
## 1180559 AX-89778325 3 426083586  
## 1180560 AX-89778326 7 666072491  
## 1180563 AX-89778329 4 68411991

On les trie dans l’ordre des chromosomes et de leur positions en bp depuis le télomère du bras court.

map <- map[order(map$Pos), ]  
map <- map[order(map$Chr), ]  
head(map)

## SNP Chr Pos  
## 524641 AX-89519629 1 45012  
## 1071353 AX-89726564 1 98264  
## 345944 AX-89445498 1 100862  
## 247647 AX-89400264 1 151786  
## 828617 AX-89635280 1 152065  
## 303025 AX-89423540 1 152603

tail(map)

## SNP Chr Pos  
## 614821 AX-89556573 14 777503232  
## 452971 AX-89489367 14 777547462  
## 957509 AX-89680733 14 777547777  
## 713797 AX-89591720 14 777548581  
## 1019901 AX-89707787 14 777549644  
## 941809 AX-89673999 14 777552033

On peut libérer de la place mémoire.

rm(genot, genot.imp, maf, p, q)

# Choix de la variable à analyser

La variable importante est placée dans la variable y qui sera ensuite manipulée.

## Exploration des relations

Ici il n’y a qu’une variable, c’est le numéro 2 (la première est le nom de la lignée)

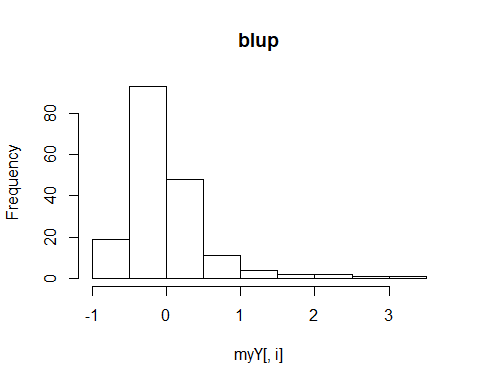
names(myY)

## [1] "Taxa" "blup" "blue"

i<-2

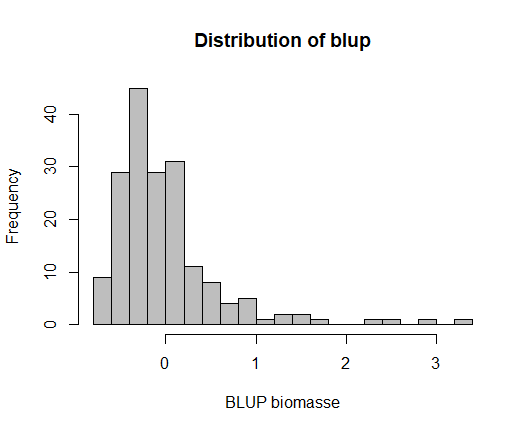
## préparation du vecteur phénotype

hist(myY[,i], main = names(myY)[i])



y <- myY[,i]  
  
# passage des noms dans le rowname de y  
names(y) <- myY$Taxa

# pour G y[which(y> 0.10)]<-NA  
  
hist(y, col = "grey", nclass=20, xlab = "BLUP biomasse", main=paste("Distribution of", names(myY)[i]))  
  
# on élimine deux données (arbitraire) qui ont des valeurs extrêmes  
#y[which(y> 0.10)]<-NA  
  
hist(y, col = "grey", nclass=20, xlab = "BLUP biomasse", main=paste("Distribution of", names(myY)[i]))



summary(y)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -0.714156 -0.355857 -0.154154 -0.001829 0.126770 3.370548

# Data merge and viz

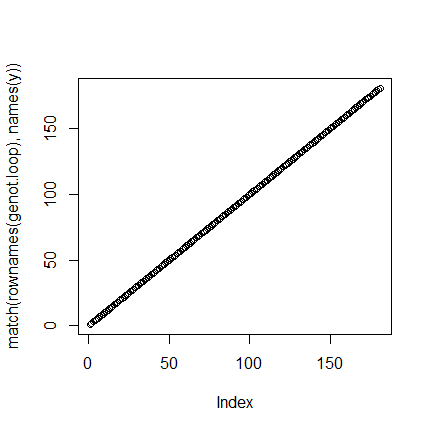
## Data check

# tri des genotypes pour qu'il soient dans le meme ordre  
genot.ok<-genot.ok[order(rownames(genot.ok)),]  
y<-y[order(names(y))]  
  
genot.loop<-genot.ok  
  
# Comme il peut y avoir des données phénotypiques manquantes  
if (length(which(is.na(y)))>0 ) {   
 liste<- which(is.na(y))  
 y <- y[-liste]   
 genot.loop<-genot.loop[-liste,]  
 #rownames(genot.loop)<-rownames(genot.ok)[-which(is.na(y))]  
 }  
  
dim(genot.loop)

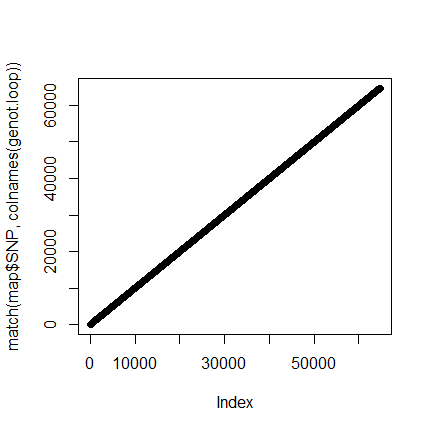
## [1] 181 64713

On verifie que les deux fichiers sont dans le meme ordre et qu’il y en a autant, ca doit faire une droite

plot(match(rownames(genot.loop), names(y)))



# tri des marqueurs pour qu'ils soient dans l'ordre de la carte  
# ne pas le faire pour avoir tous les marqueurs  
genot.loop<-genot.loop[,map$SNP]  
plot(match(map$SNP, colnames(genot.loop)))



## Calcul de la matrice de Kinship

### K Van Raden

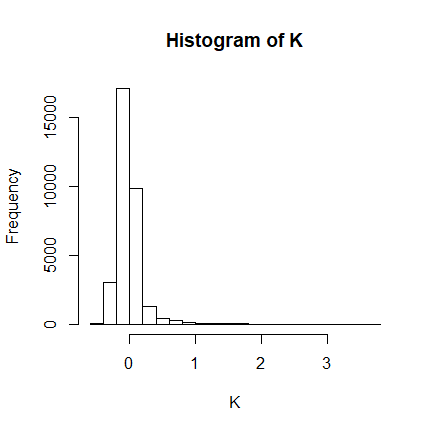
Il s’agit du calcul de la matrice d’apparentement du modèle. Il y a beaucoup de littérature sur la meilleure méthode de calcul de cette matrice. ici on utilise une méthode dite de Van Raden, pas expliquée ici.

# Calcul de la matrice d'apparentement  
p <- colMeans(genot.loop) / 2  
q <- 1 - p  
  
# Ici ca sera une Van Raden centrée  
genot.scaled <- scale(genot.loop, center = 2 \* p, scale = sqrt(2 \* p \* q))  
  
K <- tcrossprod(genot.scaled) / ncol(genot.scaled)  
K <- make.positive.definite(K)

C’est une matrice carrée qui donne les relations deux à deux pour toutes les lignées. ici elle est 176 x 176.

Voilà la distribution de toutes les valeurs. La plupart des lignées ont un très faible apparentement.

hist(K)



dim(K)

## [1] 181 181

# La régression

On prend une approche de regression multiple proposée par V.Segura et collaborateurs en utilisant le package mlmm.gwas. Mais il y a eu de très nombreuses variantes.

## Utilisation de mlmm.gwas

Vous reconnaitrez dans l’appel de la fonction  
**mlmm\_allmodels(y, list(genot.scaled), list(K), maxsteps = 4, threshold=1e-5)** les différents membres du modèle. y le phénotype à analyser, K la matrice de Kinship, genot.scaled la matrice de tous les X, Cette fonction va donc calculer en une seule fois toutes les pvalues de tous les marqueurs. Elle va aussi utiliser un niveau de risque controlé pour les tests multiples avec threshold (vous pouvez le changer) maxsteps donne le nombre de co facteurs (marqueurs les plus significatifs) qu’il va entrer

# dapres https://cran.r-project.org/web/packages/mlmm.gwas/vignettes/gwas-manual.html  
  
# dans mlmm.gwas la matrice des genotypes doit etre centrée (c'est pas écrit qu'elle doit être reduite ? )   
genot.scaled <- scale(genot.loop, center = 2 \* p)   
  
mygwas.gwas <- mlmm\_allmodels(y, list(genot.scaled), list(K),   
 maxsteps = 4, threshold=1e-5)

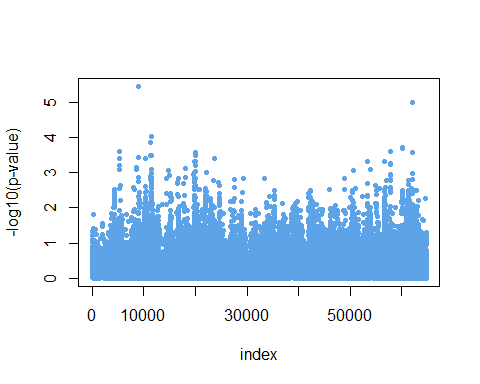
## iteration LogLik wall cpu(sec) restrained  
## 1 -89.7162 9:51:12 0 0  
## 2 -89.6008 9:51:12 0 0  
## 3 -89.5484 9:51:12 0 0  
## 4 -89.5382 9:51:12 0 0  
## 5 -89.5376 9:51:12 0 0  
## null model done! pseudo-h= 0.092   
## iteration LogLik wall cpu(sec) restrained  
## 1 -79.3956 9:51:22 0 0  
## 2 -78.6903 9:51:22 0 0  
## 3 -78.5447 9:51:22 0 0  
## 4 -78.543 9:51:22 0 0  
## 5 -78.5429 9:51:22 0 0  
## step 1 done! pseudo-h= 0.036 model: Y ~ mu + AX\_89312681   
## iteration LogLik wall cpu(sec) restrained  
## 1 -71.3066 9:51:33 0 0  
## 2 -69.7149 9:51:33 0 1  
## 3 -69.323 9:51:33 0 1  
## 4 -69.323 9:51:33 0 1  
## step 2 done! pseudo-h= 0 model: Y ~ mu + AX\_89312681 + AX\_89346847

res\_mlmm <- mygwas.gwas

Les pValues ont été calculées selon les 4 modèles et pour tous les marqueurs. L’exploitation des résultats n’est pas des plus faciles…

Pour commencer,un graphe des -log10(Pvalues) pour tous les marqueurs, quelquesoit leur position. Est ce qu’il y en a qui dépassent le seuil ?

# manhattab plot sans carte  
manhattan.plot(res\_mlmm)



Ensuite il y a une procédure (complexe) de choix du meilleur modèle.

# Selection du modèle  
sel\_XX = frommlmm\_toebic(list(genot.scaled), res\_mlmm)  
res\_eBIC = eBIC\_allmodels(y, sel\_XX, list(K), ncol(genot.scaled))  
  
res\_eBIC

## BIC ajout eBIC\_0.75 LogL  
## mu 527.2261 0.00000 527.2261 -255.8153  
## AX-89312681 510.2779 16.61658 526.8945 -244.7419  
## AX-89346847 496.6108 32.19341 528.8043 -235.3092

Le meilleur modèle est celui qui a le plus petit eBIC. Ici il n’accepte aucun SNP significatif mais on va continuer quand même.

On augmente un peu le risque de première espèce.

res\_threshold <- threshold\_allmodels(threshold=1e-4, res\_mlmm)

## Warning in rbind(res, c(names(tab), as.vector(tab), (i - 1))): number of columns  
## of result is not a multiple of vector length (arg 2)

Voilà le SNP qui passe la barre

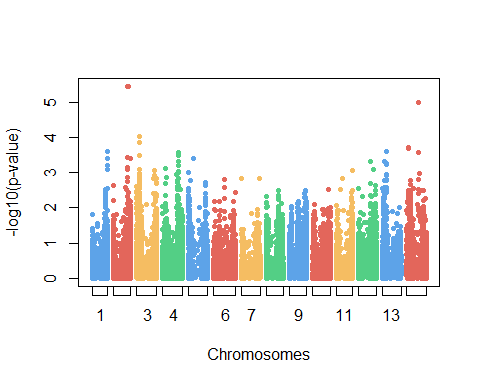
res\_threshold

## SNP p-value MLMM\_Step  
## 1 AX-89312681 AX-89493607 3.57574485007076e-06  
## 2 AX-89346847 2.29666869014092e-05 2

## Manhattan plot

On peut maintenant regarder comment se distribuent les pvalues sur le génome physique.

mip<-map  
mip$Chr<-sprintf("%2d",mip$Chr)  
  
manhattan.plot(res\_mlmm, map = mip, steps = 1, hideCofactors = FALSE, chrToPlot = "all", unit = "bp")

 Il y a un pic sur le chromosome 13 (donc le 7A).

# visualisation des effets

On peut aussi calculer les effets alléliques (la valeur de la pente)

sel\_XXclass = fromeBICtoEstimation(sel\_XX, res\_eBIC, res\_threshold)  
  
effects = Estimation\_allmodels(y, sel\_XXclass, list(K))  
effects

## BLUE Tukey.Class Frequency  
## AX-89312681\_00 0.82235207 a 0.05524862  
## AX-89312681\_11 0.00000000 b 0.94475138  
## AX-89346847\_00 0.00000000 b 0.92817680  
## AX-89346847\_11 0.70459286 a 0.07182320  
## mu -0.09786868 <NA> NA

Ici le génotype 00(aa) a une différence de -0.032 avec le génotype 11 (AA).

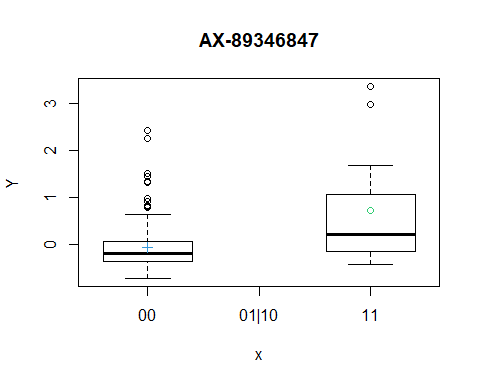
Voilà son nom.

mark\_list<-names(sel\_XXclass)  
mark\_list[1]

## [1] "AX-89312681"

Graphiquement,

# m numero du marqueur dans la liste des retenus  
m<-2  
genotypes.boxplot(genot.scaled, y , mark\_list[m], effects)



Enfin il est souvent intéressant de savoir quels sont les effectifs de chaque classe.

# les effectifs   
table(genot.loop[,mark\_list[m]])

##   
## 0 2   
## 168 13

Où est-il ?

On utilise les données connues de position

map[which(map$SNP == mark\_list[m]),]

## SNP Chr Pos  
## 86044 AX-89346847 14 475054104

A t il des marqueurs proches qui sont liés aussi ?

## En DL ?

C<-cor(genot.ok[,mark\_list[m]], genot.ok)\*\*2  
  
liste<-which(C>0.5)  
liste<-colnames(C)[liste]  
  
map[which(map$SNP %in% liste),]

## SNP Chr Pos  
## 227625 AX-89396172 14 472987161  
## 86044 AX-89346847 14 475054104  
## 481800 AX-89503308 14 476304045  
## 1086479 AX-89735540 14 479968930  
## 11693 AX-89312494 14 479971048  
## 638349 AX-89565707 14 479998922  
## 699288 AX-89584107 14 480071131  
## 475176 AX-89500220 14 480076761  
## 699037 AX-89583947 14 480515457