R

#all the washing variables are xxx1 and all the filtering variables are xxx2

setwd("/Users/celestecohen/Downloads")

#This would be Code/QTL mapping on the drive

d<-read.table(file = 'SupplementaryDataset\_S7\_genotype.tsv', sep = '\t', header = TRUE)

library(RFQTL)

#Below is formatting of data to match the formatting in the RFQTL tutorial which is online

genotype1<-d[,-c(1:4)]

genotype1<-data.matrix(genotype1)

genotype1<-t(genotype1)

phenotype1<-read.csv('Biofilm\_bioinformatics/Biofilm\_image\_processing/washing\_phenotypes.csv')

phenotype1<-phenotype1[,-1]

phenotype1<-phenotype1[which(phenotype1$strain %in% rownames(genotype1)),]

strainNames1 <- phenotype1$strain

phenotype1 <- phenotype1$ratio

sampleInfo1 <- sapply(strainNames1,FUN=function(x){

which(rownames(genotype1)==x)

})

mappingData1 <- preMap(genotype=genotype1,

phenotype=phenotype1,

sampleInfo=sampleInfo1,

scale=T)

library(randomForest)

#Get accurate realscores by calculating them 20x and averaging them – this can take 1-2h

RS1=c()

for(i in 1:20){

r=rfMapper(mappingData = mappingData1,

permute = F,

nforest = 100,

ntree = 150)

RS1=c(RS1,r)

}

RS1\_df=matrix(RS1, ncol=20, byrow=F)

RS1\_df=data.frame(RS1\_df)

realScores1<-rowMeans(RS1\_df)

#Get accurate p-values by doing 3500 permutations (100x35) – this takes 8-9h

#this wd should be an empty folder where the permutations will be stored

setwd("/Users/celestecohen/Downloads/Biofilm\_bioinformatics/washing\_permutations")

for(i in 1:35){

permutedScores1 <- rfMapper(mappingData = mappingData1,

permute = T,

nforest = 100,

ntree = 150,

nPermutations=100,

file=paste("wash\_permut",i,".RData",sep=""),

nCl=4,

clType="SOCK")

}

#this wd should be the folder in which the permutations folder is

setwd("/Users/celestecohen/Downloads/Biofilm\_bioinformatics")

#the path should be the permutations folder

#pValues are the p-values of regions of SNPs that are always grouped

pValues1 <- pEst(path="washing\_permutations/",

scores=realScores1,

markersPerIteration = 350,

printProg = T,

pCorrection = "none")

#pValuesX are the p-values for each SNP

pValuesX1 <- pValues1[mappingData1$genotype2group]

#These steps (until markerPositions) are not useful for the way I presented the results

chrVec1 <- d$chromosome

QTL\_list1 <- QTLgrouper(pmat = pValuesX1,

sigThreshold = 0.1,

corThreshold = 0.8,

distThreshold = 9,

genotype = genotype1,

chrVec = chrVec1)

markerPositions1 <- d[,c(1,2)]

markerPositions1$chromosome<-gsub("chromosome\_1",1,markerPositions1$chromosome)

markerPositions1$chromosome<-gsub("chromosome\_2",2,markerPositions1$chromosome)

markerPositions1$chromosome<-gsub("chromosome\_3",3,markerPositions1$chromosome)

markerPositions1[,3]=markerPositions1$position

#The 3 steps below are also not useful

writeQTL(QTLlist = QTL\_list1,traitNames = "Flocc",markerPositions = markerPositions1,path="myResults1.qtl")

qtl1 <- readQTL(path = "myResults1.qtl")

qtl1

#Presenting results

#barplot

barplot(-log10(pValuesX1))

abline(h=-log10(0.01),col="red")

#attempt adjusting p values afterwards, but still better without

padj=p.adjust(pValuesX1,method="fdr",n=length(pValuesX1))

barplot(as.numeric(p.adj))

#below are further attempts at FDR correction that I don’t think worked

p.tab=matrix(c(seq(1,4481,1), pValuesX1), ncol=2, byrow=F)

p.tab=data.frame(p.tab)

library(fuzzySim)

p.adj=FDR(pvalues=p.tab,q=0.01)

head(p.adj$exclude)

barplot(-log10(p.adj$exclude[,2]))

abline(h=-log10(0.01),col="red")

#manhattan plot

results.tab=markerPositions1[,-3]

results.tab[,3]=pValuesX1

colnames(results.tab)=c("CHR","BP","P")

library(qqman)

results.tab=results.tab[-which(is.na(results.tab)),]

results.tab$CHR=as.numeric(unlist(results.tab$CHR))

results.tab[,4]=rep("snp",length(results.tab$CHR))

colnames(results.tab)[4]="SNP"

manhattan(results.tab,ylim = c(0,4), suggestiveline = F, genomewideline = F,cex = 0.7)

abline(h=-log10(0.01),col="red")

text(x=1,y=-log10(0.01)+0.1,labels=" p=0.01",col="red",cex=0.7)

#from the following, determine the regions to use for reg\_map below

markerPositions1[which(-log10(pValuesX1)>=2),]

#mapping regions on chromosomes using the regions found above

#map chr

chromosomes=c("chr1","chr2","chr3")

start=c(1,1,1)

end=c(5579133,4539804,2452883)

chr\_map=matrix(c(chromosomes, start, end), ncol=3, byrow=F)

chr\_map=data.frame(chr\_map)

#map regions

regions=c("1","2","3","4","5") #this just needs to count the number of regions for formatting

chr\_names=c("chr1","chr2","chr2","chr2","chr3")

reg\_start=c(2055773,2202682,2334988,2516209,210341)

reg\_end=c(2055773,2202682,2363654,2640722,210364)

reg\_map=matrix(c(regions,chr\_names, reg\_start, reg\_end), ncol=4, byrow=F)

reg\_map=data.frame(reg\_map)

write.table(chr\_map,"chr\_map.txt",sep="\t",col.names=F,row.names=F)

write.table(reg\_map,"reg\_map.txt",sep="\t",col.names=F,row.names=F)

library(chromoMap)

chromoMap("chr\_map.txt","reg\_map.txt",segment\_annotation=T)