Estimating permutation p-values using MatrixEQTL

In our pipeline we first reformat the data per gene and then for each preprocessed gene run step4\_MatrixEQTL script which runs multiple bootstraps.

step4\_submitMatrixEQTL.R will call step4\_MatrixEQTL.R with several options: chromosome (in this example 9), number of samples inthe dataset (this can be taken from specification file) random seed 1565691 window - 5e+05 is used in this example and which model is used - shorter model for this example and optional parameter - how much paralellization you want to introduce (if your cluster supports submitting multiple jobs, for this example set to 1 meaning that every job will be run sequentially)

We load the data for MatrixEQTL

args = c("9", "704", "1565691", "5e+05", "short", "1")  
args

## [1] "9" "704" "1565691" "5e+05" "short" "1"

chri = as.numeric(args[1])  
nsub = as.numeric(args[2])  
seedval = as.numeric(args[3])  
cis\_window = as.numeric(args[4])  
model = args[5]  
if(length(args)>5){  
 paral = as.numeric(args[6])  
}else{  
 paral = 1e6  
}  
  
specf = "specifications.txt"  
  
specs = unlist(read.table(specf, as.is=T))  
pref = specs[1]  
nsam = specs[2]  
queue = specs[3]  
days = specs[4]  
days = 2  
bmem = as.numeric(specs[5])  
mem = "4g"  
if(length(specs)>20){  
 mem = specs[21]  
}  
mem

## V121   
## "4g"

seedval = specs[13]  
wrk.dir = specs[14]  
lib.dir = specs[15]  
bas.dir = specs[16]  
eigenMTdir = specs[9]  
rprog = specs[19];rprog

## V119   
## "R"

pyth = specs[20];pyth

## V120   
## "python"

setwd(wrk.dir)  
wrk.dir

## V114   
## "C:/Users/Vasyl/Documents/GitHub/asSeq/pipeline\_GTEx/v8/example/Muscle\_Skeletal"

library(MatrixEQTL)  
library(Matrix)  
useModel = modelLINEAR;   
source(sprintf("%s/helpers.R", lib.dir))  
  
  
numpoints = 100  
maf = 0.05  
   
set.seed(seedval)  
  
  
routdir = sprintf("%s/rout\_%s", wrk.dir, pref)  
boutdir = sprintf("%s/bout\_%s", wrk.dir, pref)  
if(!file.exists(routdir))dir.create(routdir)  
if(!file.exists(boutdir))dir.create(boutdir)  
  
int.dir = sprintf("%s\_%s\_%s", pref, nsub, cis\_window)  
  
  
cnt.dir = sprintf("%s\_prepr", pref);cnt.dir;file.exists(cnt.dir)

## [1] "Muscle\_Skeletal\_prepr"

## [1] TRUE

out.dir = sprintf("oneperm\_%s\_%s\_%s\_%s", pref, nsub, cis\_window, model)  
perm.dir = sprintf("boot\_%s\_%s\_%s\_%s\_%s", pref, nsub, cis\_window, model, numpoints)  
if(!file.exists(out.dir))dir.create(out.dir)  
if(!file.exists(perm.dir))dir.create(perm.dir)

Once initial setup is done we read relevant (multigene) data

genepos\_file\_name = sprintf("%s/geneInfo\_prepr\_%s.txt", cnt.dir, model)  
geneInfo = read.table(genepos\_file\_name,   
 header = T, as.is = T)  
genepos = geneInfo[geneInfo$chr==sprintf("chr%s", chri),1:4]  
genepos[,2] = gsub("chr", "", genepos[,2])  
for(coli in 3:4)genepos[,coli] = as.numeric(genepos[,coli])  
genepos

## Name chr start end  
## 15011 ENSG00000181404.15 9 14521 25004

covariates\_file\_name = sprintf("%s/Xmat\_%s.csv", int.dir, model)   
covar = read.csv(covariates\_file\_name, as.is=T, header=F)  
covar = as.matrix(covar)  
  
converge = 1e-4  
vari = apply(covar,2,var)  
  
updvar = which(vari<converge)  
for(i in updvar){  
 if(length(vari[-updvar]>0)>0){  
 correct = sqrt(median(vari[-updvar]))/sqrt(vari[i])  
 }else{  
 correct = 1/sqrt(vari[i])  
 }   
 xm = mean(covar[,i])   
 covar[,i] = xm+(covar[,i]-xm)\*correct  
}

Load gene specific data

blocki = 1  
  
 suff0 = sprintf("%s\_%s", chri, blocki)  
 timout = sprintf("%s/time\_%s.csv", perm.dir, suff0)  
   
 output\_file\_name = sprintf("%s/output\_norm\_%s.txt", int.dir, suff0)  
 output\_file\_name2 = sprintf("%s/output\_eigenMT\_%s.txt", out.dir, suff0)  
 expression\_file\_name = sprintf("%s/GE\_norm\_%s\_%s.dat", int.dir, model, suff0)  
 output\_file\_name\_min = sprintf("%s/output\_norm\_min\_%s.txt", perm.dir, suff0)  
  
 genotype\_file\_name = sprintf("%s/genotypes\_%s.dat", int.dir, suff0)  
 cvrt = SlicedData$new()  
 cvrt = cvrt$CreateFromMatrix(t(covar))  
   
 g.ini = read.table(genotype\_file\_name, header=T)  
 g.ini[g.ini==3] = 1  
 g.ini[g.ini==4] = 2  
 snpspos\_file\_name = sprintf("%s/genotypei\_%s.dat", int.dir, suff0)  
 snpspos = read.table(snpspos\_file\_name, header=T, as.is=T)  
 for(coli in 3:3)snpspos[,coli] = as.numeric(snpspos[,coli])  
 rownames(g.ini) = snpspos[,1]  
  
 kp = rowMeans(g.ini)/2  
   
 converge=5e-5  
 varZ = apply(g.ini, 1, var)  
 wVar = (varZ >= converge)  
 kp = wVar #& ((a0&a1)|(a2&a1)|(a0&a2))  
   
   
 g.ini = read.table(genotype\_file\_name, header=T)  
 g.ini[g.ini==3] = 1  
 g.ini[g.ini==4] = 2  
 snpspos\_file\_name = sprintf("%s/genotypei\_%s.dat", int.dir, suff0)  
 snpspos = read.table(snpspos\_file\_name, header=T, as.is=T)  
 for(coli in 3:3)snpspos[,coli] = as.numeric(snpspos[,coli])  
 rownames(g.ini) = snpspos[,1]  
  
 kp = rowMeans(g.ini)/2  
   
 converge=5e-5  
 varZ = apply(g.ini, 1, var)  
 wVar = (varZ >= converge)  
 kp = wVar   
  
 exprj = read.table(expression\_file\_name)  
   
   
 pvOutputThreshold = 1;  
 errorCovariance = numeric();  
   
 snps = SlicedData$new();  
 snps$fileSliceSize = 2000; # read file in pieces of 2,000 rows  
 snps = snps$CreateFromMatrix(as.matrix(g.ini))  
   
 genepos\_file\_name = sprintf("%s/genepos\_%s.dat", int.dir, suff0)  
 colnames(snpspos) = c("snpid", "chr", "pos")  
 colnames(genepos) = c("geneid", "chr", "left", "right")  
 write.table(genepos[blocki,], file=genepos\_file\_name, row.names=F, col.names=T, quote=F, sep="\t")  
  
 rownames(exprj) = genepos$geneid[blocki]  
 gene = SlicedData$new();  
 gene = gene$CreateFromMatrix(as.matrix(exprj))

Load information for the relevant chromosome

genepos\_file\_name = sprintf("%s/geneInfo\_prepr\_%s.txt", cnt.dir, model)  
geneInfo = read.table(genepos\_file\_name,   
 header = T, as.is = T)  
genepos = geneInfo[geneInfo$chr==sprintf("chr%s", chri),1:4]  
genepos[,2] = gsub("chr", "", genepos[,2])  
for(coli in 3:4)genepos[,coli] = as.numeric(genepos[,coli])  
genepos

## Name chr start end  
## 15011 ENSG00000181404.15 9 14521 25004

covariates\_file\_name = sprintf("%s/Xmat\_%s.csv", int.dir, model)   
covar = read.csv(covariates\_file\_name, as.is=T, header=F)  
covar = as.matrix(covar)  
  
converge = 1e-4  
vari = apply(covar,2,var)  
  
updvar = which(vari<converge)  
for(i in updvar){  
 if(length(vari[-updvar]>0)>0){  
 correct = sqrt(median(vari[-updvar]))/sqrt(vari[i])  
 }else{  
 correct = 1/sqrt(vari[i])  
 }   
 xm = mean(covar[,i])   
 covar[,i] = xm+(covar[,i]-xm)\*correct  
}

Load gene specific data

blocki = 1  
countjobs = 0  
  
 suff0 = sprintf("%s\_%s", chri, blocki)  
 timout = sprintf("%s/time\_%s.csv", perm.dir, suff0)  
   
 output\_file\_name = sprintf("%s/output\_norm\_%s.txt", int.dir, suff0)  
 output\_file\_name2 = sprintf("%s/output\_eigenMT\_%s.txt", out.dir, suff0)  
 expression\_file\_name = sprintf("%s/GE\_norm\_%s\_%s.dat", int.dir, model, suff0)  
 output\_file\_name\_min = sprintf("%s/output\_norm\_min\_%s.txt", perm.dir, suff0)  
  
 genotype\_file\_name = sprintf("%s/genotypes\_%s.dat", int.dir, suff0)  
 cvrt = SlicedData$new()  
 cvrt = cvrt$CreateFromMatrix(t(covar))  
   
 g.ini = read.table(genotype\_file\_name, header=T)  
 g.ini[g.ini==3] = 1  
 g.ini[g.ini==4] = 2  
 snpspos\_file\_name = sprintf("%s/genotypei\_%s.dat", int.dir, suff0)  
 snpspos = read.table(snpspos\_file\_name, header=T, as.is=T)  
 for(coli in 3:3)snpspos[,coli] = as.numeric(snpspos[,coli])  
 rownames(g.ini) = snpspos[,1]  
  
 kp = rowMeans(g.ini)/2  
   
 converge=5e-5  
 varZ = apply(g.ini, 1, var)  
 wVar = (varZ >= converge)  
 kp = wVar #& ((a0&a1)|(a2&a1)|(a0&a2))  
   
   
 g.ini = read.table(genotype\_file\_name, header=T)  
 g.ini[g.ini==3] = 1  
 g.ini[g.ini==4] = 2  
 snpspos\_file\_name = sprintf("%s/genotypei\_%s.dat", int.dir, suff0)  
 snpspos = read.table(snpspos\_file\_name, header=T, as.is=T)  
 for(coli in 3:3)snpspos[,coli] = as.numeric(snpspos[,coli])  
 rownames(g.ini) = snpspos[,1]  
  
 kp = rowMeans(g.ini)/2  
   
 converge=5e-5  
 varZ = apply(g.ini, 1, var)  
 wVar = (varZ >= converge)  
 kp = wVar   
  
 SNP\_file\_name = sprintf("%s/SNP\_%s.txt", int.dir, suff0)  
  
 write.table(g.ini, SNP\_file\_name, row.names=T, col.names=T, quote=F, sep="\t")  
  
 exprj = read.table(expression\_file\_name)  
   
   
 pvOutputThreshold = 1;  
 errorCovariance = numeric();  
   
 snps = SlicedData$new();  
 snps$fileSliceSize = 2000; # read file in pieces of 2,000 rows  
 snps = snps$CreateFromMatrix(as.matrix(g.ini))  
   
 genepos\_file\_name = sprintf("%s/genepos\_%s.dat", int.dir, suff0)  
 colnames(snpspos) = c("snpid", "chr", "pos")  
 colnames(genepos) = c("geneid", "chr", "left", "right")  
 write.table(genepos[blocki,], file=genepos\_file\_name, row.names=F, col.names=T, quote=F, sep="\t")  
  
 rownames(exprj) = genepos$geneid[blocki]  
 gene = SlicedData$new();  
 gene = gene$CreateFromMatrix(as.matrix(exprj))

Initial MatrixEQTL run

output\_file\_name

## [1] "Muscle\_Skeletal\_704\_5e+05/output\_norm\_9\_1.txt"

me = Matrix\_eQTL\_main(  
 snps = snps,  
 gene = gene,  
 cvrt = cvrt,  
 pvOutputThreshold = 1e-200,  
 output\_file\_name = sprintf("%s\_tmp", output\_file\_name),  
 output\_file\_name.cis = output\_file\_name,  
 pvOutputThreshold.cis = pvOutputThreshold,  
 useModel = useModel,   
 errorCovariance = errorCovariance,  
 snpspos = snpspos,  
 genepos = genepos[blocki,],   
 cisDist = 1e9,  
 verbose = TRUE,  
 pvalue.hist = TRUE,  
 min.pv.by.genesnp = FALSE,  
 noFDRsaveMemory = FALSE);

## Matching data files and location files

## 1 of 1 genes matched

## 1613 of 1613 SNPs matched

## Task finished in 0.02 seconds

## Processing covariates

## Task finished in 0.02 seconds

## Processing gene expression data (imputation, residualization)

## Task finished in 0 seconds

## Creating output file(s)

## Task finished in 0.03 seconds

## Performing eQTL analysis

## 100.00% done, 1,613 cis-eQTLs, 0 trans-eQTLs

## No significant associations were found.  
## 5

## Task finished in 0.5 seconds

##

file.remove(sprintf("%s\_tmp", output\_file\_name))

## [1] TRUE

names(me)

## [1] "time.in.sec" "param" "all" "trans" "cis"

eigenMT correction

cmdi = sprintf("%s %s/eigenMT.py --CHROM %s --QTL %s --GEN %s --GENPOS %s --PHEPOS %s --OUT %s",   
 pyth, eigenMTdir, chri, output\_file\_name, SNP\_file\_name, snpspos\_file\_name,   
 genepos\_file\_name, output\_file\_name2)  
 message(cmdi)

## python C:/Users/Vasyl/Documents/GitHub/asSeq/pipeline\_GTEx/v8/example/lib/eigenMT/eigenMT.py --CHROM 9 --QTL Muscle\_Skeletal\_704\_5e+05/output\_norm\_9\_1.txt --GEN Muscle\_Skeletal\_704\_5e+05/SNP\_9\_1.txt --GENPOS Muscle\_Skeletal\_704\_5e+05/genotypei\_9\_1.dat --PHEPOS Muscle\_Skeletal\_704\_5e+05/genepos\_9\_1.dat --OUT oneperm\_Muscle\_Skeletal\_704\_5e+05\_short/output\_eigenMT\_9\_1.txt

system(cmdi)

Run permutation estimate (calling newscript runboot to produce 1000 iterations for 100 points) with the refitting on the same data MatrixEQTL) Note, here we disabled submission to the cluster, so example gene will be run directly on the local machine.

eigenMT = read.table(output\_file\_name2, header=T, as.is=T)  
 m = match(eigenMT$SNP, snpspos$snpid)  
 eigenMT$chr = chri  
 eigenMT$snppos = snpspos$pos[m]  
 eigenMT$genestart = genepos$left[blocki]  
 eigenMT$geneend = genepos$right[blocki]  
 #get minimum p-values and respective snps  
 genes=as.character(me$cis$eqtls$gene)  
 ords = data.frame(t(sapply(sort(unique(genes)), minord, genes=me$cis$eqtls$gene, pvals=me$cis$eqtl$pvalue)))  
 ords[,2] = as.numeric(as.character(ords[,2]))  
 pvals = aggregate(me$cis$eqtls$pvalue, by=list(me$cis$eqtls$gene), FUN=min)  
 table(pvals[,2]==me$cis$eqtl$pvalue[ords[,2]])

##   
## TRUE   
## 1

ords$betas=me$cis$eqtl$beta[ords[,2]]  
 ords$tstat=me$cis$eqtls$statistic[ords[,2]]  
 ords$pvals=me$cis$eqtls$pvalue[ords[,2]]  
 ords$snps=me$cis$eqtls$snps[ords[,2]]  
 ords

## X1 X2 betas tstat pvals  
## ENSG00000181404.15 ENSG00000181404.15 1 0.9185603 14.08248 8.582963e-40  
## snps  
## ENSG00000181404.15 chr9\_520337\_T\_G\_b38

ntest = aggregate(rep(1, length(me$cis$eqtls$pvalue)), by=list(me$cis$eqtls$gene), FUN=sum)  
 m = match(ords[,1], ntest[,1])  
 table(ntest[m,1]==ords[,1])

##   
## TRUE   
## 1

ords$ntest = ntest[m,2]  
   
 nmedp = aggregate(me$cis$eqtls$pvalue, by=list(me$cis$eqtls$gene), FUN=median)  
 m = match(ords[,1], nmedp[,1])  
 table(nmedp[m,1]==ords[,1])

##   
## TRUE   
## 1

ords$nmedp = nmedp[m,2]  
   
 m = match(ords[,1], eigenMT$gene)  
 m

## [1] 1

table(ords[,1] == eigenMT$gene[m])

##   
## TRUE   
## 1

ords$TESTS = eigenMT$TESTS[m]  
 eigenMT$ntest[m] = ords$ntest  
 eigenMT = eigenMT[m,]  
   
 #need to refit minimums with linear model to get other covariates  
 m = match(eigenMT$SNP, rownames(g.ini))  
 table(eigenMT$SNP==rownames(g.ini)[m])

##   
## TRUE   
## 1

gen.sub = matrix(g.ini[m,],nrow=length(m));rownames(gen.sub) = rownames(g.ini)[m]  
 colnames(gen.sub) = colnames(g.ini)  
 write.csv(gen.sub, sprintf("%s/min\_snp\_vals\_%s.csv", out.dir, suff0), quote=F, row.names=F)  
 table(rownames(gen.sub)==eigenMT$SNP)

##   
## TRUE   
## 1

write.csv(eigenMT, sprintf("%s/upd\_eigenMT\_%s.csv", out.dir, suff0), quote=F, row.names=F)  
   
   
 filout = sprintf("%s/short\_pval\_%s.csv", out.dir, suff0)  
 write.table(ords[,-c(2)], filout, sep=",", row.names=F, col.names=T)  
   
   
   
 #write intermediate objects  
 write.csv(exprj, sprintf("%s/expr\_%s.csv", out.dir, suff0), quote=F)  
 write.csv(gen.sub, sprintf("%s/msnp\_%s.csv", out.dir, suff0), quote=F)  
   
 rinpdir = lib.dir  
 rinp = sprintf("%s/step4\_runboot.R", rinpdir)  
 rout = sprintf("%s/step4\_runboot\_%s\_%s\_%s\_%s\_%s\_%s\_%s.Rout",  
 routdir, chri, blocki, nsub, numpoints, cis\_window, model, seedval)   
 qout = sprintf("%s/step4\_runboot\_%s\_%s\_%s\_%s\_%s\_%s\_%s.out",  
 boutdir, chri, blocki, nsub, numpoints, cis\_window, model, seedval)   
 rprog = "R"  
 com = sprintf("%s CMD BATCH \"--args %s %s %s %s %s %s %s\" %s %s",  
 rprog, chri, blocki, nsub, numpoints, cis\_window, model, seedval, rinp, rout)  
 com2 = sprintf("sbatch -p %s -t 0%s-00:00:00 -o %s --mem=%s --wrap='%s\'",   
 queue, days, qout, mem, com)   
 if(blocki%%paral==0){  
 message(com)  
 system(com)  
 }else{  
 message(com2)  
 system(com2)  
 }

## R CMD BATCH "--args 9 1 704 100 5e+05 short 1565691" C:/Users/Vasyl/Documents/GitHub/asSeq/pipeline\_GTEx/v8/example/lib/step4\_runboot.R C:/Users/Vasyl/Documents/GitHub/asSeq/pipeline\_GTEx/v8/example/Muscle\_Skeletal/rout\_Muscle\_Skeletal/step4\_runboot\_9\_1\_704\_100\_5e+05\_short\_1565691.Rout

Lets illustrate calculation of permutation p-value estimate. We take the values generated in step4\_runboot.R and fit glm predicting probability of observing more extreme result (then observed in bootstrap) by log10(minimum p-value). After fitting glm, predict permutation p-value based on log10(minimum p-value) Effective number of tests will be ratio of predicted permutation p-value and minimum p-value (trimmed between 1 and number of SNPs)

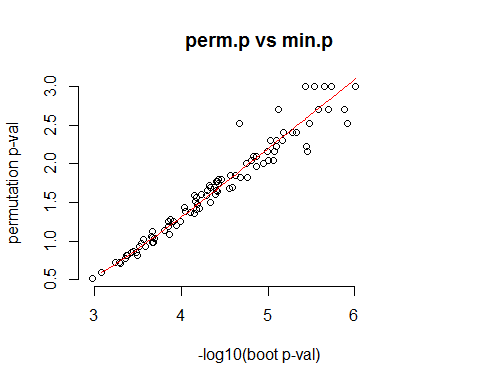
boots = read.csv(sprintf("%s/short\_boot\_pval\_9\_1.csv", perm.dir), as.is=T)  
eigenMT = read.csv(sprintf("%s/upd\_eigenMT\_9\_1.csv", out.dir), as.is=T)  
nperm = 1000  
y = boots$permp\*nperm  
pvalb = boots$pvalb  
kp3 = (y/nperm)>=0 & (y/nperm)<=0.3  
kp3a = (y/nperm)>0 & (y/nperm)<=0.3  
   
y1 = log10(y/nperm)  
x1 = log10(pvalb)  
glmi3 = glm(cbind(y[kp3],nperm-y[kp3])~x1[kp3], family="binomial")  
summary(glmi3)

##   
## Call:  
## glm(formula = cbind(y[kp3], nperm - y[kp3]) ~ x1[kp3], family = "binomial")  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.2386 -0.7024 -0.0964 0.6726 2.3370   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 5.32658 0.12261 43.45 <2e-16 \*\*\*  
## x1[kp3] 2.07455 0.03285 63.16 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 6555.17 on 98 degrees of freedom  
## Residual deviance: 102.31 on 97 degrees of freedom  
## AIC: 559.99  
##   
## Number of Fisher Scoring iterations: 4

xval = log10(eigenMT$p.value)  
pred.perm = logiti(glmi3$coef[1]+glmi3$coef[2]\*xval)  
c(xval, pred.perm)

## (Intercept)   
## -3.906636e+01 1.305708e-33

xlim = range(-c(x1, xval))  
ylim = range(-log10(y/nperm))  
ylim[2] = -log10(pred.perm)  
  
plot(-x1, -log10(y/nperm), xlab="-log10(boot p-val)", ylab="permutation p-val", bty="n", main="perm.p vs min.p")  
o = order(x1[kp3])  
xf = x1[kp3][o]  
yf = glmi3$fitted.values[o]  
lines(-xf, -log10(yf), col="red")



fit = seq(0, xval, length.out=50)  
pred.perm0 = logiti(glmi3$coef[1]+glmi3$coef[2]\*fit)  
plot(-x1, -log10(y/nperm), xlab="-log10(boot p-val)", ylab="permutation p-val", bty="n", main="perm.p vs min.p", xlim=xlim,ylim=ylim)  
lines(-fit, -log10(pred.perm0), col="red")  
points(-xval, -log10(pred.perm), col="blue", cex=1, pch=19)  
legend("topleft", "estimated permu.p", text.col="blue", pch=19, col="blue", bty="n")

