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

```
getwd()

## [1]
"C:/Users/Vasyl/Documents/GitHub/asSeq/pipeline_GTEx/v8/example/Muscle_Skelet
al"

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"
getwd()

## [1]
"C:/Users/Vasyl/Documents/GitHub/asSeq/pipeline_GTEx/v8/example/Muscle_Skelet
al"

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_Skelet
al"

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)
```

```r
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
```

```r
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

```r
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
```

```r
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

```r
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

```
getwd()

## [1]
"C:/Users/Vasyl/Documents/GitHub/asSeq/pipeline_GTEx/v8/example/Muscle_Skelet
al"

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 seconds

## Processing gene expression data (imputation, residualization)

## Task finished in 0.01 seconds

## Creating output file(s)

## Task finished in 0.04 seconds

## Performing eQTL analysis

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

## No significant associations were found.
## 5

## Task finished in 0.54 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/ei
genMT.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_runb
oot.R
C:/Users/Vasyl/Documents/GitHub/asSeq/pipeline_GTEx/v8/example/Muscle_Skeleta
l/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)
```

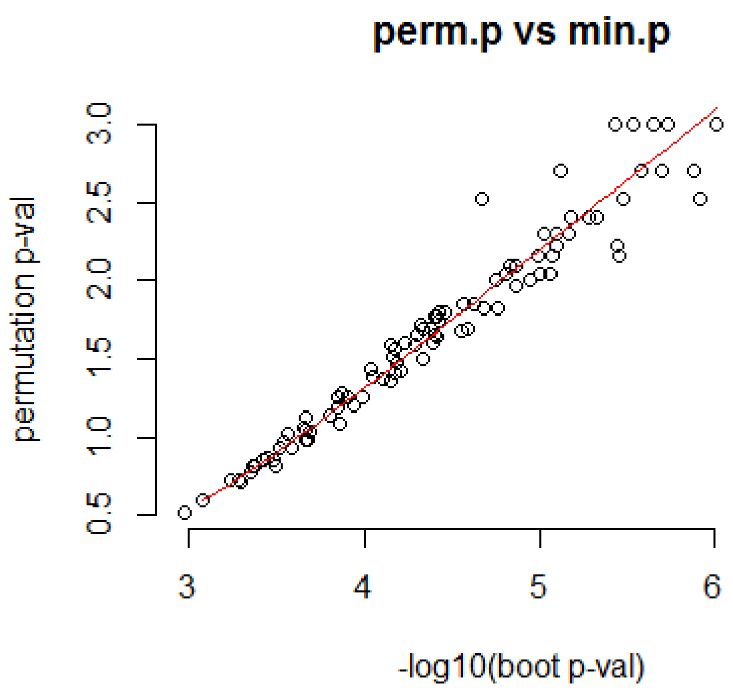
```
##
## Call:
## glm(formula = cbind(y[kp3], nperm - y[kp3]) ~ x1[kp3], family =
"binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2386  -0.7024  -0.0964   0.6726   2.3370
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  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)
```

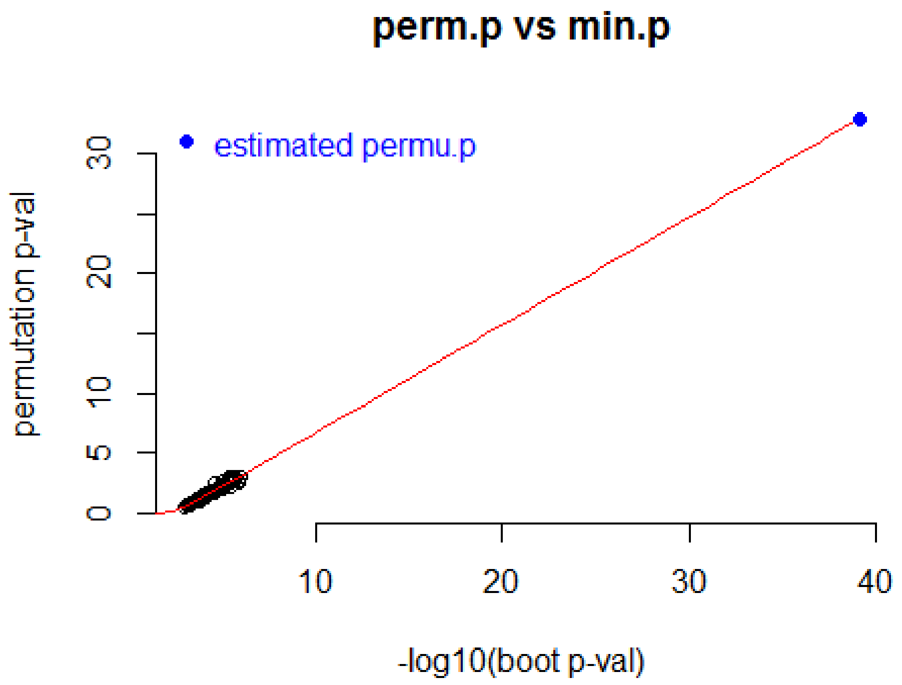
```
##                  (Intercept)
## -3.906636e+01  1.305708e-33
```

```
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")
```


perm.p vs min.p
● estimated permu.p
permutation p-val
0
5
10
20
30
10
20
30
40
-log10(boot p-val)