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.

First, load the data for MatrixEQTL. Arguments here are in the same style as original pipeline script. They give information about the chromosome on which gene is located, number of subsamples to be used for estimation (no more then total number of samples recorded in specification file), random seed, window size and which model to be used.

Specification file still will be used, since it is required at earlier steps linking in this pipeline. It is not necessary if you choose a different way to provide the path to the data.

The data in this example is simulated based on the GTEx dataset which allows to avoid distribution of the real data, but provides a dataset that is well represented of the GTEx dataset.

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

Load genotype data. Check that there is no snps with too small variance.

Load geneinfo and design matrix. Ensure that covariates don't have too small variance. Load gene expression (already quantile normalized format provided in the example).

```
genepos_infile_name = sprintf("%s/%s/geneInfo_prepr_%s.txt", bas.dir, cnt.dir, model)
geneInfo = read.table(genepos_infile_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])
colnames(genepos) = c("geneid", "chr", "left", "right")

genepos_file_name = sprintf("%s/genepos_%s.dat", int.dir, suff0)
colnames(snpspos) = c("snpid", "chr", "pos")
write.table(genepos[blocki,], file=genepos_file_name,
            row.names=F, col.names=T, quote=F, sep="\t")

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
}
cvrt = SlicedData$new()
cvrt = cvrt$CreateFromMatrix(t(covar))


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();

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]
```

Permutation estimate rewritten as a function.

If you ran eigenMT prior to running this pipeline, you can use that result to get a better guess of effective number of tests but it can be skipped. In such case 1/4 of simple number of SNPs can be used as a proxy for initial guess of effective number of SNPs. The procedure will then continue the adjustment if initial guess is too inconsistent based on first X iterations.

Among the outputs procedure will give estimated permutation p-value and implied effective number of tests in fields pred.permGLM and TESTSupd of the summ object.

```
#will now create an object which would contained required information
permEst = list(snpM=as.matrix(g.ini[,subs]),
               geneM=as.matrix(exprj[,subs]),
               cvrtM=as.matrix(covar[subs,]),
               snpspos=snpspos,
               genepos=genepos,
               outpf=sprintf("%s_mEQTL_unr.txt", rownames(exprj)[1]),
               pvOutputThreshold=1e-300,
               pvOutputThreshold.cis=1,
               cisDist=1e9,#we have already preprocessed SNPs
               errMumGuess=nrow(g.ini)/4,
               verbose=FALSE, pvalue.hist=FALSE,
               min.pv.by.genesnp = FALSE,
               noFDRsaveMemory=FALSE,
               outdir="unreduced")
#updNtests=sprintf("%s_updtests.csv", rownames(exprj)[1])
me = getPermP(permEst, n.perm=100, ini.perm=25)
```

```
me$summ$pred.permGLM
```

```
## [1] 1.154033e-10
```

```
me$summ$TESTSupd
```

```
## [1] 1613
```

Above mentioned run was done for the case with very significant minimum p-value, for which it is less critical to establish permutation p-value, since it will be significant using various methods of estimating permutation p-value (for example, a quick tool to do this is eigenMT)

What happens if we have less significant p-value. Consider a modification of the defined above gene expression data with scaled down effect size. We can produce a bootstrap with reduced effect size using get_reduced_bood function and repeat the procedure based on this reduced expression.

Note, that this step includes a target permutation p-value that is guessed based on a guess of effective number of tests, so the resulting p-value can be somewhat outside of the target range.

```
#now, same but imagine reduced effect size
permEst$outpf=sprintf("%s_mEQTL_red.txt", rownames(exprj)[1])

eigenMT = me$summ
eigenMT$TESTS = eigenMT$TESTSupd
gen.sub = me$min.snp
redboot = get_reduced_boot(1, target.perm.ps=5e-2,
                           mQTL.fit=eigenMT,
                           expr.mat = permEst$geneM,
                           min.SNP=gen.sub,
                           covars=permEst$cvrtM,
                           nsam=ncol(gen.sub))

permEstR = permEst
permEstR$geneM = redboot
rownames(permEstR$geneM) = rownames(permEst$geneM)

permEstR$effNumGuess=eigenMT$TESTSupd
permEstR$outdir="reduced"

meR = getPermP(permEstR, n.perm=100, ini.perm=25)
```

```
meR$summ$pred.permGLM
```

```
## [1] 0.001759128
```

```
meR$summ$TESTSupd
```

```
## [1] 1613
```

Lets illustrate calculation of permutation p-value estimate based on the reduced bootstrap. We take the values generated in the prevous step 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) using less extreme effect size:

```
nperm = 100
eigenMT = meR$summ
y = meR$vals$ermp*nperm
pvalb = meR$vals$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
## -2.3390  -0.7035  -0.0545   0.4775   2.3340
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.38179    0.26385    20.4   <2e-16 ***
## x1[kp3]      2.08628    0.07346    28.4   <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: 1478.05  on 178  degrees of freedom
## Residual deviance:  164.86  on 177  degrees of freedom
## AIC: 695.61
##
## Number of Fisher Scoring iterations: 5
```

```
xval = log10(eigenMT$p.value)
pred.perm = logit1(glmi3$coef[1]+glmi3$coef[2]*xval)
c(xval, pred.perm)
```

```
##                (Intercept)
## -5.619083198  0.001759128
```

```
xlim = range(-c(x1, xval))
kp = y!=0
ylim = range(-log10(y/nperm)[kp])
ylim[2] = max(c(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", ylim=ylim)
o = order(x1[kp3])
xf = x1[kp3][o]
yf = glmi3$fitted.values[o]
lines(-xf, -log10(yf), col="red")
points(-xval, -log10(pred.perm), col="blue", cex=1, pch=19)

fit = seq(0, min(c(xval,x1)), length.out=50)
pred.perm0 = logit1(glmi3$coef[1]+glmi3$coef[2]*fit)
lines(-fit, -log10(pred.perm0), col="red")
```