Estimating permutation p-values using MatrixEQTL

The basic pipeline includes data preparation steps:

1 step0_submit_trim.R - trimming suspicious allele-specific counts and total counts 2 step1_submit_preprocSNP.R - creating separate files for each gene

main eQTL analysis and p-value estimate:

3 step2_submit_trecaseA.R 4 step4_submit_MatrixEQTL.R

and further analysis steps including principal component analysis and dynamic eqtl findigs.

Once the specification file is set up, each script should be able to be run authomatically (with corresponding note given to the script to distinguish between long and short model, etc)

1 Exact code to be run:

R CMD BATCH '–args specifications_Muscle_Skeletal.txt long' step0_submit_trim.R MS_step0_submit_trim_long.Rout

1.a To save time short model fit relies on long model being run first, so this step should be done after the previous is completed.

R CMD BATCH '–args specifications_Muscle_Skeletal.txt short' step0_submit_trim.R MS_step0_submit_trim_short.Rout 2 wait until the previous step completes

R CMD BATCH '–args specifications_Muscle_Skeletal.txt' step1_submit_preprocSNP.R MS_step1_submit_preprocSNP.Rout 3 wait until the previous step completes fitting steps can be run at once, if you have generous enough scheduler (often schedulers have limit on the number of maximum number of jobs that can be submitted)

fitting TReCASE model for each gene, long model:

R CMD BATCH '–args specifications_Muscle_Skeletal.txt long 5e5' step2_submit_trecaseA.R MS_step2_submit_trecaseA_long.Rout

R CMD BATCH '–args specifications_Muscle_Skeletal.txt short 5e5' step2_submit_trecaseA.R MS_step2_submit_trecaseA_short.Rout

4 estimating permuted p-values

R CMD BATCH '–args specifications_Muscle_Skeletal.txt long 5e5' step4_submit_MatrixEQTL.R MS_step4_submit_MatrixEQTL_long.Rout

R CMD BATCH '–args specifications_Muscle_Skeletal.txt short 5e5' step4_submit_MatrixEQTL.R MS_step4_submit_MatrixEQTL_short.Rout

Assuming, that previous step reformatting the data is finished, step4_MatrixEQTL script will submit each gene to produce runs multiple bootstraps to estimate permutation p-value.

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)

```
com = sprintf("R CMD BATCH  \"--args specifications_Muscle_Skeletal.txt short 5e5\" step4_submit_MatrixEQTL.R MS_step4_submit_MatrixEQTL_short.Rout")
message(com)

## R CMD BATCH  "--args specifications_Muscle_Skeletal.txt short 5e5" step4_s
ubmit_MatrixEQTL.R MS_step4_submit_MatrixEQTL_short.Rout

system(com)
```

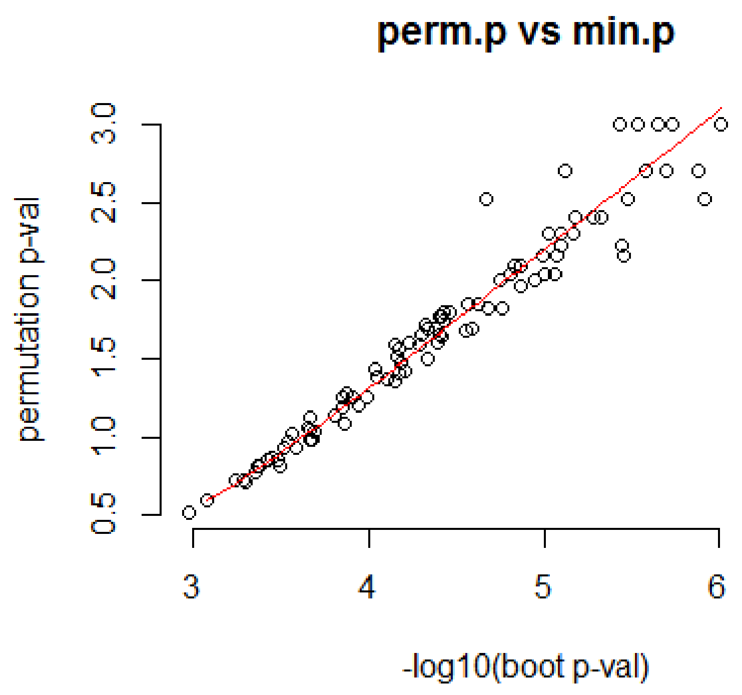
Lets illustrate calculation of permutation p-value estimate. We take the values generated in step4_runboot.R and fit glm predicting probability of observing more extreme result (then observed in bootstrap) by log10(minimum p-value). After fitting glm, predict permutation p-value based on log10(minimum 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)

```
setwd("C:/Users/Vasyl/Documents/GitHub/asSeq/pipeline_GTEx/v8/example/Muscle_Skeletal")
perm.dir = "boot_Muscle_Skeletal_704_5e+05_short_100"
out.dir = sprintf("oneperm_Muscle_Skeletal_704_5e+05_short")
logiti = function(x)1/(1+exp(-x))


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
```

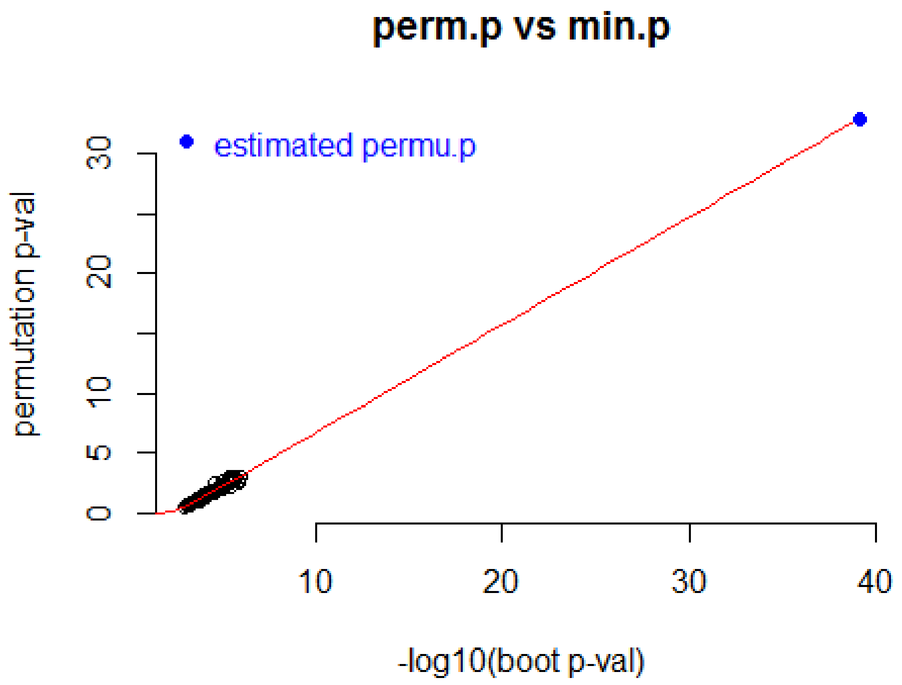
```
xval = log10(eigenMT$p.value)
pred.perm = logiti(glmi3$coef[1]+glmi3$coef[2]*xval)
c(xval, pred.perm)
```

```
##                  (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")
```



```r
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", b
ty="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)