

eQTL analysis to identify sex-by-SNP interactions

Warren Anderson
Joon Yuhl Soh
Mete Civelek

April 3, 2020

This guide provides code and documentation of analyses from Anderson et al., 2020, *Sex differences in human adipose tissue gene expression and genetic regulation involve adipogenesis*

Contents

1	Overview	2
2	Preparing a SNP matrix file	2
3	Identifying an optimal number of PEER factors	3
4	Correction of the GTEx data for eQTL analysis using latent factors identified by PEER	11
5	Implement the eQTL analysis using <i>MatrixEQTL</i> in R	14
6	References	29

List of Figures

1	Number of egenes with respect to PEER count.	11
2	PCA with covariate annotation following PEER correction.	15

1 Overview

Here we show the code used to correct the expression data based on known and hidden covariates.

2 Preparing a SNP matrix file

Modify the provided vcf file to input into the eqtl function.

```
tail -n +1 GTEx_Analysis_2017-06-05_v8_WholeGenomeSeq_838Indiv_AllVar_QC_metrics.vcf > total_chroms.vcf

bcftools view -i 'INFO/R2 > 0.9' total_chroms.vcf > total_chroms_rsqr.vcf

# nano to add o in description in two header rows
head -n 91 GTEx_Analysis_2017-06-05_v8_WholeGenomeSeq_838Indiv_AllVar_QC_metrics.vcf > head.txt
nano head.txt
tail -n +1 head.txt > head.vcf
tail -n +92 GTEx_Analysis_2017-06-05_v8_WholeGenomeSeq_838Indiv_AllVar_QC_metrics.vcf >> head.vcf

vcftools --vcf head.vcf --remove-indels --maf 0.05 --hwe 0.000001 \
--max-missing 0.98 --recode --recode-INFO-all --out \
GTEx_Analysis_2017-06-05_v8_WholeGenomeSeq_838Indiv_AllVar_QC_metrics.recode.vcf

head -n 91 GTEx_Analysis_2017-06-05_v8_WholeGenomeSeq_838Indiv_AllVar_QC_metrics.recode.vcf.recode.vcf | \
tail -1 > geno_sample_id.txt

head -n 90 GTEx_Analysis_2017-06-05_v8_WholeGenomeSeq_838Indiv_AllVar_QC_metrics.recode.vcf.recode.vcf > outSNP.vcf
```

Next recreate the sampleID using R.

```
library(dplyr)
subq_id = read.table("subq_subjects.txt", sep = "\", header=F, stringsAsFactors=F) %>% t
geno = read.table("geno_sample_id.txt", sep = "\",
  header=F, stringsAsFactors=F, comment.char="") %>% t
subq_id = as.data.frame(subq_id, stringsAsFactors=F)
geno = as.data.frame(geno, stringsAsFactors=F)
names(subq_id) = names(geno) = "id"
col_subq = sapply(subq_id$id, function(x) which(geno$id==x)) %>% unlist

tc <- c(1:9, col_subq)
genoOut <- geno[tc,1]
write.table(t(genoOut), "subjList.txt", sep = "\", col.names=F, row.names=F, quote=F)
system("tail -n +1 subjList.txt >> outSNP.vcf")
tc1 <- paste(unlist(tc), collapse=",$")
tc2 <- paste("$", tc1, sep="")
tc3 <- paste("tail -n +92 recode.vcf | awk -v OFS='\"'
  'print \"\", tc2, \"\" >> outSNP.vcf", sep="")
system(tc3)
```

Now process the vcf with tabix.

```
# generate tab delimited columns and remove blank columns
head -n 91 outSNP.vcf | tail | awk 'print NF' | sort -nu | tail -n 1
head -n 91 outSNP.vcf | tail | tr " " "\\" | awk 'print NF' | sort -nu | tail -n 1
cat outSNP.vcf | tr " " "\\" > gtex_genosubq_tab0.vcf
sed 's/\\t+/\\t/g;s/\\t//' gtex_genosubq_tab0.vcf > gtex_genosubq_tab1.vcf
mv gtex_genosubq_tab1.vcf gtex_genosubq.vcf

# tabix processing
PATH=$PATH:/media/wa3j/Seagate2/Documents/software/tabix-0.2.6
bgzip -c gtex_genosubq.vcf > gtex_genosubq_indexed.vcf.gz
```

```
tabix -p vcf gtex_genotype_subq_indexed.vcf.gz
```

Here we create the snp matrix file.

```
# generate SNP matrix in R
library(dplyr)
library(VariantAnnotation)
library(snpStats)
fname = "gtex_genotype_subq_indexed.vcf.gz"
tab <- TabixFile(fname, yieldSize=500000)
param <- ScanVcfParam(genotype=c("GT"))
open(tab)
ii = 0
while (nrow(vcf_yield <- readVcf(tab, "hg19", param=param))) {
  mat <- genotypeToSnpMatrix(vcf_yield)
  newdat = t(as(mat$genotype, "numeric")) %>% as.data.frame
  if(nrow(newdat)*ncol(newdat) > 0){
    out = cbind(rownames(newdat),newdat)
    colnames(out)[1] = "id"
    if(ii==0){colTF=TRUE;apnd=FALSE}else{colTF=FALSE;apnd=TRUE}
    write.table(out,file="gtex_snp_mat.txt",append=apnd,sep="\\",
               col.names=colTF,row.names=F,quote=F)
  }
  ii=1
}
close(tab)

# checks
hdr <- scanVcfHeader(fname)
names(info(hdr))
head(genotype(vcf_yield)$GT[,1:6])
head(newdat[,1:6])
```

3 Identifying an optimal number of PEER factors

We define the optimal PEER factor count based on maximization of eQTL interactions detected at a given significance level. Here we vary the number of PEER factors and evaluate the number of eQTL interactions detected over a range of association p-values. We move the following files onto a server for the analysis: *covar_eqtl.txt*, *expr_eqtl.txt*, *gtex_snp_mat.txt*, *genecoordmat.txt*, and *snpcoordmat.txt*. The following R script, saved as *optimize_peers.R* implements the basic eQTL analysis for a given number of peer factors. Following the R script, we show the shell script that calls the *optimize_peers.R* from separate cores, one per peer factor included in the analysis. The file *peer_count.txt* should be established to contain a column vector of numbers for the peer counts. We evaluate 2, 4, 6, ..., 40 peer factors (n=20). Details of each step of the following script will be explained following the code.

```
cat > optimize_peers.R <<'EORS'
library(dplyr)
library(MatrixEQTL)
library(peer)
write.table(seq(2,40,2),"peer_count.txt",quote=F,col.names=F,row.names=F)

#####
## import data
#####
```

```

# number of peers to consider
args = commandArgs(trailingOnly=TRUE)

fname = "subq_gtex_invNorm.txt"
expr0 = read.table(fname,stringsAsFactors=F,header=T,sep="\t",check.names=F)
fname = "genes_XYM.txt"
genes_XYM = read.table(fname,header=F,sep="\t",stringsAsFactors=F)
fname = "gencode_gene_map.txt"
ann_gene0 = read.table(fname,header=T,sep="\t",stringsAsFactors=F)
fname = "gtex_subq_covars.txt"
demo00 = read.table(fname,header=T,sep="\t",stringsAsFactors=F)
fname = "Adipose_Subcutaneous.v8.covariates.txt"
ecov0 = read.table(fname,header=T,sep="\t",stringsAsFactors=F,check.names=F) %>% t

names = ecov0[1,]
ecov0 = ecov0[-1,] %>% as.data.frame
names(ecov0) = names

ecov0Index <- match( demo00$SUBJID, rownames(ecov0))
ecov1 <- ecov0[ecov0Index,]
demo0 <- demo00
demo0$C1 <- ecov1$PC1
demo0$C2 <- ecov1$PC2
demo0$C3 <- ecov1$PC3
demo0[,69:71] = apply(demo0[,69:71],2,function(c){data.matrix(c) %>% as.numeric})

# set data organization
all(demo0$SAMPID == names(expr0))

# get gene annotation for expressed transcripts
inds = sapply(rownames(expr0),function(x){which(ann_gene0$gene_id==x)}) %>% unlist
gene.map = ann_gene0[inds,]
all(gene.map$gene_id == rownames(expr0))

# omit transcripts on sex chromosomes
ind.sex = sapply(genes_XYM[,1],function(x){which(gene.map$gene_name==x)}) %>% unlist
expr1 = expr0[-ind.sex,]
gene.map = gene.map[-ind.sex,]

# peer numbers
fname = "peer_count.txt"
peer0 = read.table(fname,stringsAsFactors=F,header=F,sep="\t",check.names=FALSE)
n.peers = peer0[args[1],1]
print(paste0("the number of peer factors is ",n.peers))

#####
## regress expression data against covariates
#####

# regress out covariates to be corrected explicitly covariate
# model_pr = t(covar0[,-1]) %>% as.data.frame
# names(model_pr) = covar0[,1]
# model_pr = model_pr %>% select(Platform,C1,C2,C3,SMRIN,AGE,GENDER)
model_pr = demo0 %>% select(Platform,C1,C2,C3,SMRIN,AGE,GENDER)
model_pr$Platform = as.factor(model_pr$Platform)
model_pr$GENDER = as.factor(model_pr$GENDER)
# fitdat = expr0[,2:ncol(expr0)] %>% t

```

```

fitdat = expr1 %>% t
fit = apply(fitdat,2,function(x){
  dat = as.data.frame(cbind(x,model_pr))
  fit_regdat = lm(x ~ .,data=dat)
  return(fit_regdat$residuals)
})
resids = t(fit)
# rownames(resids) = expr0$id

#####
## implement peer for a given number of factors
#####

peer_model = PEER()
PEER_setPhenoMean(peer_model, t(resids))
PEER_setAdd_mean(peer_model, TRUE)
PEER_setNk(peer_model,n.peers)
PEER_update(peer_model)
peers = PEER_getX(peer_model)
# peer_out = peers[,2:ncol(peers)]

#####
## combine peer factors and covariates for matricexeqtl
#####

# annotate peer factors
peer_out = peers[,2:ncol(peers)]
colnames(peer_out) = paste0("peer",c(1:n.peers))
rownames(peer_out) = names(expr1)

# aggregate peers and covariate data
covar_out = cbind(model_pr,peer_out)

# combine covariates with peer factors
# peer.names = paste0("peer",1:n.peers)
# peer.mat = cbind(peer.names,t(peer_out)) %>% as.data.frame(stringsAsFactors=FALSE)
# names(peer.mat) = names(covar0)
# names(peer.mat) = c("id", names(expr1))
# cdat1 = cbind(names(model_pr),t(model_pr)) %>% as.data.frame(stringsAsFactors=FALSE)
# names(cdat1) = c("id", names(expr1))
# covar.mat = rbind(peer.mat,cdat1)
fout = paste0("peercovar_",n.peers,".txt")
write.table(covar_out,fout,col.names=T,row.names=F,sep="\t",quote=F)

system("head -1 gtex_snp_mat.txt > genoNames.txt")

# import genotype identifier list
fname = "genoNames.txt"
geno.name = read.table(fname,header=F,stringsAsFactors=F) %>% t
geno.name = geno.name[-1,] %>% as.character

# verify organization the expression and covariate matrices
samp.rename = function(ids=NULL){
  out = sapply(ids,function(x){
    out1 = strsplit(x,"-")[[1]]
    out2 = paste0(out1[1],"-",out1[2])
    return(out2)
  }) %>% unlist

```

```

    return(out)
}
expr.ids = samp.rename(names(expr1))
geno.ids = samp.rename(geno.name)
all(expr.ids == geno.ids)

# replace the identifiers in the expression/covariate frames
# with those from the genotype file
eqtl.id.map = cbind(geno.ids,names(geno.ids),names(expr.ids)) %>% as.data.frame
names(eqtl.id.map) = c("subject_id","geno_id","expr_id")
rownames(eqtl.id.map) = 1:nrow(eqtl.id.map)
expr_eqtl = expr1
names(expr_eqtl) = eqtl.id.map$geno_id
cv = covar_out
cv$GENDER = as.character(covar_out$GENDER)
covar_eqtl = t(cv)
colnames(covar_eqtl) = eqtl.id.map$geno_id

# reorganize covariates so that the interaction term is last
# recode sex to 0/m, 1/f
numberOfR <- nrow(covar_eqtl)
org.inds = c(1:6,8:numberOfR,7)
covar_eqtl = covar_eqtl[org.inds,]
covar_eqtl[numberOfR,which(covar_eqtl[numberOfR,]=="Male")] = 0
covar_eqtl[numberOfR,which(covar_eqtl[numberOfR,]=="Female")] = 1
covar_eqtl = covar_eqtl %>% as.data.frame

# format and output covariate/expression data for MatrixEQTL
expr_eqtl = cbind(rownames(expr_eqtl),expr_eqtl) %>% as.data.frame
names(expr_eqtl)[1] = "id"
covar_eqtl = cbind(rownames(covar_eqtl),covar_eqtl) %>% as.data.frame
names(covar_eqtl)[1] = "id"
write.table(expr_eqtl,"expr_eqtl.txt",col.names=T,row.names=F,sep="\t",quote=F)
write.table(covar_eqtl,"covar_eqtl.txt",col.names=T,row.names=F,sep="\t",quote=F)

system("cat gtex_snp_mat.txt | cut -f1 > snp_ids.txt")

# get SNP coordinates
fname = "snp_ids.txt"
geno.id = read.table(fname,stringsAsFactors=F,header=T)
chrs = paste0("chr",sapply(geno.id$id,function(x){strsplit(x,"_")[[1]][1]}))
pos = sapply(geno.id$id,function(x){strsplit(x,"_")[[1]][2]})
snpcoord = cbind(geno.id$id,chrs,pos) %>% as.data.frame
snpcoord$pos = snpcoord$pos %>% data.matrix %>% as.numeric
names(snpcoord) = c("snp","chr","pos")
rownames(snpcoord) = 1:nrow(snpcoord)

# get gene coordinates
gene_anno = read.table("gencode.txt",header=F,sep="\t",stringsAsFactors=F)
geneNames = sapply(gene_anno[,5],function(x){
  out1 = strsplit(x," ")[[1]]
  ind = grep("gene_id",out1)
  out2 = out1[ind]
  out3 = strsplit(out1[ind+1],";")[[1]][1]
  out = c(out2,out3)
  return(out)
}) %>% t
rownames(geneNames) = c(1:nrow(geneNames))

```

```

unique(geneNames[,1])
gene_info = cbind(geneNames[,2],gene_anno[,c(1,3,4)]) %>% as.data.frame
names(gene_info) = c("geneid","chr","s1","s2")
gene_info$chr = sapply(gene_info$chr,function(x){paste0("chr",x)})
expr.inds = match(rownames(expr_eqtl),gene_info$geneid)
genecoords = gene_info[expr.inds,]

# output genomic coordinates
write.table(snpcoord,"snpcoordmat.txt",sep="\t",quote=F,row.names=F,col.names=T)
write.table(genecoords,"genecoordmat.txt",sep="\t",quote=F,row.names=F,col.names=T)

#####
## parameters for matrixeqtl
#####
base.dir = getwd()
useModel = modelLINEAR_CROSS

# Genotype information
SNP_file_name = "gtex_snp_mat.txt"
snps_location_file_name = "snpcoordmat.txt"

# Gene expression information
expression_file_name = "expr_eqtl.txt"
gene_location_file_name = "genecoordmat.txt"

# Covariates information
covariates_file_name = "covar_eqtl.txt"

# Distance for local gene-SNP pairs
cisDist = 1e6

#####
## implement matrixeqtl
#####

## Load genotype data
snps = SlicedData$new();
snps$fileDelimiter = "\t";
snps$fileOmitCharacters = "NA";
snps$fileSkipRows = 1;
snps$fileSkipColumns = 1;
snps$fileSliceSize = 2000;
snps$LoadFile(SNP_file_name);

## Load gene expression data
gene = SlicedData$new();
gene$fileDelimiter = "\t";
gene$fileOmitCharacters = "NA";
gene$fileSkipRows = 1;
gene$fileSkipColumns = 1;
gene$fileSliceSize = 2000;
gene$LoadFile(expression_file_name);

## Load covariates
cvrt = SlicedData$new();
cvrt$fileDelimiter = "\t";
cvrt$fileOmitCharacters = "NA";
cvrt$fileSkipRows = 1;

```

```

cvrt$fileSkipColumns = 1;
if(length(covariates_file_name)>0) {
  cvrt$LoadFile(covariates_file_name);
}

# load genomic coordinates data
snpspos = read.table(snps_location_file_name, header = TRUE, stringsAsFactors = FALSE)
genepos = read.table(gene_location_file_name, header = TRUE, stringsAsFactors = FALSE)

# run analysis
filename = tempfile(pattern="file", tmpdir=base.dir, fileext="")
qtl = Matrix_eQTL_main(
  snps = snps,
  gene = gene,
  cvrt = cvrt,
  output_file_name = "",
  pvOutputThreshold = 0,
  useModel = useModel,
  errorCovariance = numeric(),
  verbose = TRUE,
  output_file_name.cis = filename,
  pvOutputThreshold.cis = 1,
  snpspos = snpspos,
  genepos = genepos,
  cisDist = cisDist,
  pvalue.hist = "qqplot",
  min.pv.by.genesnp = FALSE,
  noFDRsaveMemory = FALSE);
unlink( filename )

# output results
dat = qtl$cis$eqtls
fout = paste0("peercovar_",n.peers,"ciseqtl.txt")
imout = paste0("peercovar_",n.peers,"image.RData")

print(head(dat))
print(fout)

save.image(imout)
write.table(dat,fout,col.names=T,row.names=F,sep="\t",quote=F)

EORS

```

The analysis is implemented in parallel using the following shell script, *peer_slurm.sh*.

```

cat > peer_slurm.sh <<'EOS'
#!/bin/bash
#SBATCH -c 8
#SBATCH -A civelekLab
#SBATCH -p standard
#SBATCH --time=0-05:30:00

module load gcc/7.1.0 openmpi
module load R/3.5.1

i=${SLURM_ARRAY_TASK_ID}
chmod 700 peer_analysis.sh
./peer_analysis.sh $i

```



```

# done
EOS

#####
## generate a script for each core
#####
cat > peer_analysis.sh <<'EOS'
#!/bin/bash
i="$1"
# go to base directory
dir=/nv/vol192/civeleklab/J00N/v8/12
cd $dir
# move files into a directory for each peer count
n0=peer_count_${i}
mkdir ${n0}
cp -t ${n0} optimize_peers.R gtex_snp_mat.txt subq_gtex_invNorm.txt genes_XYM.txt
  gencode_gene_map.txt gtex_subq_covars.txt gencode.txt \
  Adipose_Subcutaneous.v8.covariates.txt
cd ${n0}

cat > peerScript${i}.sh <<EOF
# initialize log file
exec &> log_${n0}.txt
echo $n0
echo ""
# call the R script
Rscript --vanilla optimize_peers.R ${i}
EOF

echo calling peerScript${i}.sh
chmod 700 peerScript${i}.sh
./peerScript${i}.sh

EOS

```

The script can be implemented by the following command line entry.

```

sbatch --array=1-20 peer_slurm.sh

```

We then analyze the data in R as follows. At the significance level of 10^{-6} , maximal egenes were identified for 32 PEER factors. The results are shown in Figure 1. First, import the data.

```

library(dplyr)

# files to loop through
file.prefix = "peercovar_"
files = Sys.glob(file.path("/nv/vol192/civeleklab/J00N/v8/12/peer_count_",
  paste(file.prefix, "*ciseqtl.txt", sep = '')))

# parameters and data initialization
pcut = 0.001
p.levels = c(1e-3, 1e-4, 1e-5, 3e-6, 1e-6)
eqtl.dat = c()

# loop through each file and save relevant data
for(ii in 1:length(files)) {

```

```

# get number of peer factors
split1 = strsplit(files[ii], "peercovar_")[[1]][2]
npeers = strsplit(split1,"ciseqtl.txt")[[1]][1] %>% as.numeric
print(paste0("running analysis for npeers = ",npeers))

## perform initial filter in unix
system( paste0("head -1 ",files[ii]," > tmp.head.txt") )
system( paste0("awk '$4 <= ",pcut,"' ",files[ii]," > tmp0.txt") )
system( paste0("cat tmp.head.txt tmp0.txt > tmp.txt") )

## read in the data
dat = read.table("tmp.txt",header=T,sep="\t",stringsAsFactors=F)

# filter and save data at each p-value level of interest (p.levels)
mat = c()
for(jj in 1:length(p.levels)){
  ngenes = dat %>% filter(pvalue < p.levels[jj]) %>% select(gene) %>% unique %>% nrow
  mat = rbind(mat, c(npeers,ngenes,p.levels[jj]))
}
colnames(mat) = c("npeers","ngenes","pval")

# update data storage and remove temp files
eqtlmat = rbind(eqtlmat,mat)
system("rm *tmp*")

} # ii, loop through files

eqtlmat = as.data.frame(eqtlmat)

```

Plot the result of the analysis to decide how many peer factors should be used.

```

library(ggplot2)
library(gridExtra)

eqtlmat$pval = as.factor(eqtlmat$pval)

plt = ggplot(data=eqtlmat, aes(x=npeers, y=ngenes, group=pval)) +
  geom_line(aes(color=pval)) +
  geom_point(aes(color=pval)) +
  scale_y_log10() +
  theme(text = element_text(size=20))

plts = list(plt)
pdf("vary_peer_count.pdf", onefile = FALSE)
marrangeGrob(grobs=plts, nrow=1, ncol=1, top=NULL)
dev.off()

# Find which number of peer factor is optimal
targetPG <- eqtlmat$ngenes[eqtlmat$pval == 1e-06]
targetPP <- eqtlmat$npeers[eqtlmat$pval == 1e-06]
maxNGenes <- targetPG == max(targetPG)
maxNPeers <- targetPP[maxNGenes]
print(max(targetPG)) # 56
print(maxNPeers) # 32

```

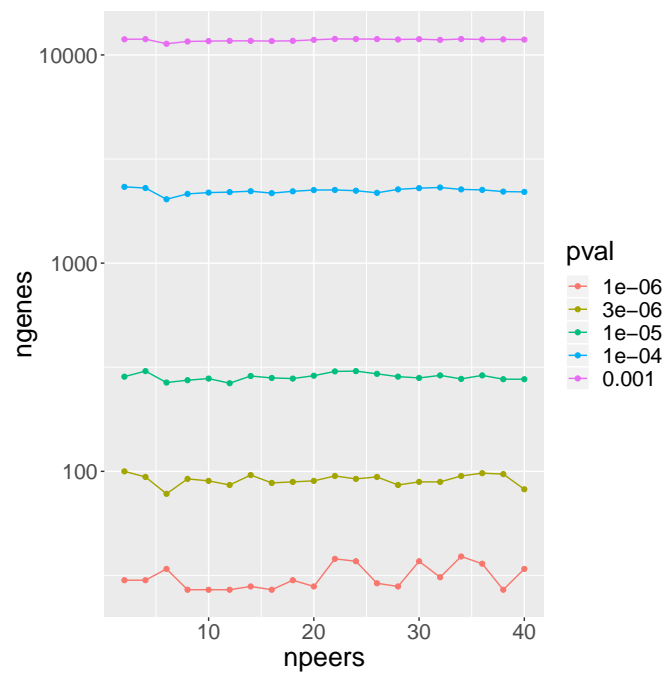


Figure 1: Number of egenes with respect to PEER count.

4 Correction of the GTEx data for eQTL analysis using latent factors identified by PEER

Procedures for processing the GTEx subcutaneous adipose data and subject/sample information (co-variates) are described in *FigS3.GTEx.pdf*. First we import the data and perform some basic processing operations to check the data organization and isolate autosomal genes.

```
library(dplyr)
library(peer)

fname = "subq_gtex_invNorm.txt"
expr0 = read.table(fname,stringsAsFactors=F,header=T,sep="\t",check.names=F)
fname = "genes_XYM.txt"
genes_XYM = read.table(fname,header=F,sep="\t",stringsAsFactors=F)
fname = "gencode_gene_map.txt"
ann_gene0 = read.table(fname,header=T,sep="\t",stringsAsFactors=F)
fname = "gtex_subq_covars.txt"
demo00 = read.table(fname,header=T,sep="\t",stringsAsFactors=F)
fname = "Adipose_Subcutaneous.v8.covariates.txt"
ecov0 = read.table(fname,header=T,sep="\t",stringsAsFactors=F,check.names=F) %>% t
names = ecov0[1,]
ecov0 = ecov0[-1,] %>% as.data.frame
names(ecov0) = names
ecov0Index <- match( demo00$SUBJID, rownames(ecov0))

ecov1 <- ecov0[ecov0Index,]
demo0 <- demo00
demo0$C1 <- ecov1$PC1
demo0$C2 <- ecov1$PC2
demo0$C3 <- ecov1$PC3
demo0[,69:71] = apply(demo0[,69:71],2,function(c){data.matrix(c) %>% as.numeric})
```

```
# set data organization
all(demo0$SAMPID == names(expr0))

# get gene annotation for expressed transcripts
inds = sapply(rownames(expr0),function(x){which(ann_gene0$gene_id==x)}) %>% unlist
gene.map = ann_gene0[inds,]
all(gene.map$gene_id == rownames(expr0))

# omit transcripts on sex chromosomes
ind.sex = sapply(genes_XYM[,1],function(x){which(gene.map$gene_name==x)}) %>% unlist
expr1 = expr0[-ind.sex,]
gene.map = gene.map[-ind.sex,]
```

Next, we explicitly correct the data for the following covariates: platform (from the GTEx eQTL covariate file), genotype PC1-3 (from the GTEx eQTL covariate file), RIN, age, and sex. We use multivariate linear regression to adjust the expression data.

```
# isolate covariates of interest for explicit correction
model_pr = demo0 %>% select(Platform,C1,C2,C3,SMRIN,AGE,GENDER)
model_pr$Platform = as.factor(model_pr$Platform)
model_pr$GENDER = as.factor(model_pr$GENDER)

# regress out covariates to be corrected explicitly covariate
fitdat = expr1 %>% t
fit = apply(fitdat,2,function(x){
  dat = as.data.frame(cbind(x,model_pr))
  fit_regdat = lm(x~.,data=dat)
  return(fit_regdat$residuals)
})
resids_pr = t(fit)
```

With the data corrected for particular covariates of interest, we now implement the PEER analysis to identify latent factors presumably unrelated to the aforementioned covariates. We parameterize the analysis to estimate 32 latent factors based on a previous analysis which showed that this number maximized the interactions detected.

```
nOfPeers <- 32

# implement PEER
peer_model = PEER()
PEER_setPhenoMean(peer_model, t(resids_pr))
PEER_setAdd_mean(peer_model, TRUE)
PEER_setNk(peer_model,nOfPeers)
PEER_update(peer_model)
peers = PEER_getX(peer_model)

# annotate peer factors
peer_out = peers[,2:ncol(peers)]
colnames(peer_out) = paste0("peer",c(1:nOfPeers))
rownames(peer_out) = names(expr1)

# aggregate peers and covariate data
covar_out = cbind(model_pr,peer_out)
```

The analysis converged after 93 iterations. Note that the analysis gave 33 columns, the first of which was all ones and was removed. Next we perform some basic quality control analyses to evaluate the features of the latent factors identified by PEER. First, we found that none of the PEER factors were correlated with either RIN or age, as should be the case given that the data subjected to the PEER

analysis were already adjusted for RIN and age. Maximal absolute Pearson correlation coefficients were ~ 0 .

```
# check correlations with RIN and age
cors.rin = apply(covar_out[8:ncol(covar_out)],2,function(x) cor(x,covar_out$SMRIN))
cors.age = apply(covar_out[8:ncol(covar_out)],2,function(x) cor(x,covar_out$AGE))
max(abs(cors.rin))
max(abs(cors.age))
```

Next, we adjusted the data for platform , genotype PC1-3, RIN, age, and 32 peers and implemented PCA to evaluate the influences of known covariates (see *FigS3.GTEx.pdf*). Note that we do not correct for sex in this analysis.

```
library(ggplot2)
library(gridExtra)

# adjust the data based on peers and known covariates
fitdat = expr1 %>% t
fit = apply(fitdat,2,function(x){
  dat = as.data.frame(cbind(x,covar_out[, -7]))
  fit_regdat = lm(x~.,data=dat)
  return(fit_regdat$residuals)
})
resids_all = t(fit)
names(resids_all) = names(expr1)

# implement PCA based on SVD
PC <- prcomp(t(resids_all),scale.=T,center=T)
pc_scores = PC$x
pc_loadings = PC$rotation
pc_eigvals = PC$sdev^2
pc_percent = 100 * pc_eigvals / sum(pc_eigvals)

# function for generating annotated PCA plots
pca.plot.fun = function(data=NULL,col=NULL,legend.dir=NULL) {
  plt.out = ggplot(data,aes_string(x="PC1",y="PC2",colour=col)) +
    scale_colour_manual(values = c("red","blue","green","cyan","magenta")) +
    geom_point(size=3,alpha=0.7) +
    theme(legend.position="top",legend.direction=legend.dir) +
    theme(plot.margin = unit( c(mr,mr,mr,mr) , "in" ) ) +
    xlab(paste0("PC1 (",round(pc_percent[1],1),"%)" ) ) +
    ylab(paste0("PC2 (",round(pc_percent[2],1),"%)" ) ) +
    stat_ellipse(type="norm",level=0.75,size=2)
  return(plt.out)
}

# plot params
mr = 0.1

# analyze RIN effects by plotting PC projection annotated according to RIN quartile
quantile_RIN = quantile(covar_out$SMRIN)
rin = covar_out$SMRIN
rin[covar_out$SMRIN==min(covar_out$SMRIN)] = 1
for(ii in 2:length(quantile_RIN)){
  ind = which(rin > quantile_RIN[ii-1] & rin <= quantile_RIN[ii])
  rin[ind] = ii-1
}
PCi = data.frame(pc_scores,RIN_quartile=as.factor(rin))
rin.plt = pca.plot.fun(data=subset(PCi,RIN_quartile==c(1,4)),
```

```

col="RIN_quartile",legend.dir="vertical")

# analyze the effects of age
covage = covar_out %>% mutate(age_class = NA)
covage$age_class[covar_out$AGE < 40] = "<40"
covage$age_class[covar_out$AGE > 60] = ">60"
PCi = data.frame(pc_scores,Age=as.factor(covage$age_class))
age.plt = pca.plot.fun(data=subset(PCi,Age==c("<40",">60")),
  col="Age",legend.dir="vertical")

# analyze the effects of death circumstances
PCi = data.frame(pc_scores,Hardy_scale=as.factor(demo0$DTHHRDY))
hrdy.plt = pca.plot.fun(data=PCi,col="Hardy_scale",legend.dir="horizontal")

# analyze the effects of sex
PCi = data.frame(pc_scores,Sex=as.factor(covar_out$GENDER))
sex.plt = pca.plot.fun(data=PCi,col="Sex",legend.dir="vertical")

# plot the covariate effects data for RIN etc
plts = list(rin.plt,hrdy.plt,age.plt,sex.plt)
pdf("PCA_covar_peer.pdf", onefile = FALSE)
marrangeGrob(grobs=plts, nrow=2, ncol=2, top=NULL)
dev.off()

```

The results are shown in Figure 2. The first two PCs captured little variability (1-3%), indicating that the PEER correction abrogated much of the existing structure in the data, as was the case for surrogate variable-based correction (see *FigS3.GTEx.pdf*). The explicit adjustment for age and RIN apparently nullified the influences of these covariates on the first PC. The effects of death circumstance (Hardy scale metric) were also removed, thereby suggesting that PEER effectively corrected for such consistent influences on the data structure. However, as the data were corrected for the categorical sex variable before implementing PEER, correcting for the PEER factors left the influence of sex both intact and poignant. Sex could account for the prominent separation of two data clouds on the first PC, indicating that sex was a variable contributing to the systematic variation in the data.

5 Implement the eQTL analysis using *MatrixEQTL* in R

First we verify that the genotype file has the proper organization corresponding to the covariate and expression matrices. In the command line we isolate the sample identifiers in the genotype file.

```
system("head -1 gtex_snp_mat.txt > genoNames.txt")
```

In R we compare the genotype organization to the organization in our expression/covariate matrices. We also use the sample identifiers from the genotype file (mainly corresponding to book samples) as the identifiers for the expression/covariate matrices for technical convenience. Basic formatting is performed and data are written to file for subsequent eQTL analysis.

```

# import genotype identifier list
fname = "genoNames.txt"
geno.name = read.table(fname,header=F,stringsAsFactors=F) %>% t
geno.name = geno.name[-1,] %>% as.character

# verify organization of the expression and covariate matrices
samp.rename = function(ids=NULL){
  out = sapply(ids,function(x){
    out1 = strsplit(x,"-")[[1]]
    out2 = paste0(out1[1],"-",out1[2])
    return(out2)
  })
}

```

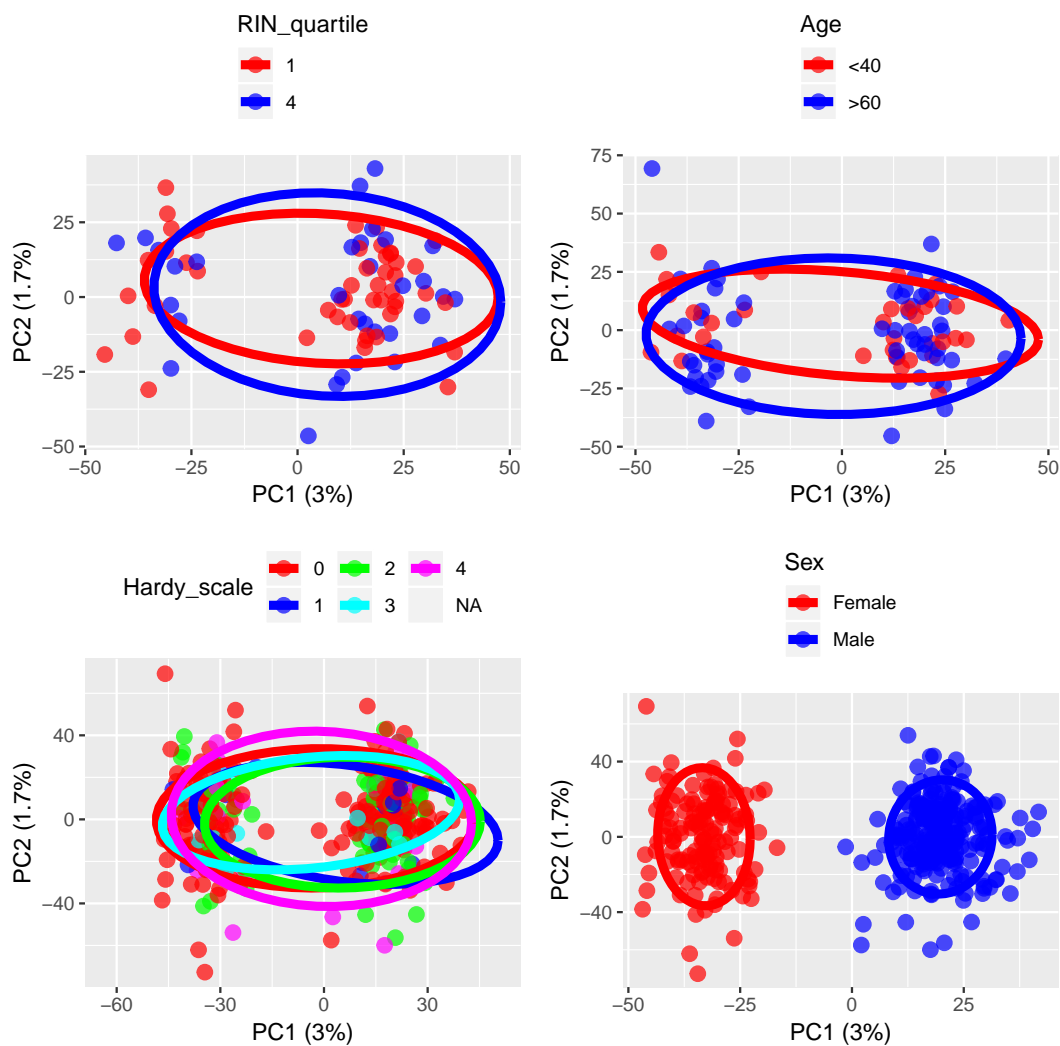


Figure 2: PCA with covariate annotation following PEER correction.

```

    }) %>% unlist
    return(out)
  }
  expr.ids = samp.rename(names(expr1))
  geno.ids = samp.rename(geno.name)
  all(expr.ids == geno.ids)

  # replace the identifiers in the expression/covariate frames
  # with those from the genotype file
  eqtl.id.map = cbind(geno.ids, names(geno.ids), names(expr.ids)) %>% as.data.frame
  names(eqtl.id.map) = c("subject_id", "geno_id", "expr_id")
  rownames(eqtl.id.map) = 1:nrow(eqtl.id.map)
  expr_eqtl = expr1
  names(expr_eqtl) = eqtl.id.map$geno_id
  cv = covar_out
  cv$GENDER = as.character(covar_out$GENDER)
  covar_eqtl = t(cv)
  colnames(covar_eqtl) = eqtl.id.map$geno_id

  # reorganize covariates so that the interaction term is last

```

```
# recode sex to 0/m, 1/f
numberOfR <- nrow(covar_eqtl)
org.inds = c(1:6,8:numberOfR,7)
covar_eqtl = covar_eqtl[org.inds,]
covar_eqtl[numberOfR,which(covar_eqtl[numberOfR,]=="Male")] = 0
covar_eqtl[numberOfR,which(covar_eqtl[numberOfR,]=="Female")] = 1
covar_eqtl = covar_eqtl %>% as.data.frame

# format and output covariate/expression data for MatrixEQTL
expr_eqtl = cbind(rownames(expr_eqtl),expr_eqtl) %>% as.data.frame
names(expr_eqtl)[1] = "id"
covar_eqtl = cbind(rownames(covar_eqtl),covar_eqtl) %>% as.data.frame
names(covar_eqtl)[1] = "id"
write.table(expr_eqtl,"expr_eqtl.txt",col.names=T,row.names=F,sep="\t",quote=F)
write.table(covar_eqtl,"covar_eqtl.txt",col.names=T,row.names=F,sep="\t",quote=F)
```

Now we must specify SNP and gene coordinates. we use the command line to get the SNP identifiers that contain the chromosomal coordinates.

```
system("cat gtex_snp_mat.txt | cut -f1 > snp_ids.txt")
```

Then we use R to get the formatted data frames.

```
# get SNP coordinates
fname = "snp_ids.txt"
geno.id = read.table(fname,stringsAsFactors=F,header=T)
chrs = paste0("chr",sapply(geno.id$id,function(x){strsplit(x,"_")[[1]][1]}))
pos = sapply(geno.id$id,function(x){strsplit(x,"_")[[1]][2]})
snpcoord = cbind(geno.id$id,chrs,pos) %>% as.data.frame
snpcoord$pos = snpcoord$pos %>% data.matrix %>% as.numeric
names(snpcoord) = c("snp","chr","pos")
rownames(snpcoord) = 1:nrow(snpcoord)

# get gene coordinates
gene_anno = read.table("gencode.txt",header=F,sep="\t",stringsAsFactors=F)
geneNames = sapply(gene_anno[,5],function(x){
  out1 = strsplit(x," ")[[1]]
  ind = grep("gene_id",out1)
  out2 = out1[ind]
  out3 = strsplit(out1[ind+1],";")[[1]][1]
  out = c(out2,out3)
  return(out)
}) %>% t
rownames(geneNames) = c(1:nrow(geneNames))
unique(geneNames[,1])
gene_info = cbind(geneNames[,2],gene_anno[,c(1,3,4)]) %>% as.data.frame
names(gene_info) = c("geneid","chr","s1","s2")
gene_info$chr = sapply(gene_info$chr,function(x){paste0("chr",x)})
expr.inds = match(rownames(expr_eqtl),gene_info$geneid)
genecoords = gene_info[expr.inds,]

# output genomic coordinates
write.table(snpcoord,"snpcoordmat.txt",sep="\t",quote=F,row.names=F,col.names=T)
write.table(genecoords,"genecoordmat.txt",sep="\t",quote=F,row.names=F,col.names=T)
```

Next we implement the cis-eQTL analysis (1 mB) with a sex \times SNP interaction model.


```

# eQTL analysis parameters/annotation
library(MatrixEQTL)
useModel = modelLINEAR_CROSS
SNP_file_name = "gtex_snp_mat.txt"
expression_file_name = "expr_eqtl.txt"
covariates_file_name = "covar_eqtl.txt"
errorCovariance = numeric()
cisDist = 1e6

## Load genotype data
snps = SlicedData$new();
snps$fileDelimiter = "\t";      # the TAB character
snps$fileOmitCharacters = "NA"; # denote missing values;
snps$fileSkipRows = 1;         # one row of column labels
snps$fileSkipColumns = 1;      # one column of row labels
snps$fileSliceSize = 2000;     # read file in slices of 2,000 rows
snps$LoadFile(SNP_file_name);

## Load gene expression data
gene = SlicedData$new();
gene$fileDelimiter = "\t";      # the TAB character
gene$fileOmitCharacters = "NA"; # denote missing values;
gene$fileSkipRows = 1;         # one row of column labels
gene$fileSkipColumns = 1;      # one column of row labels
gene$fileSliceSize = 2000;     # read file in slices of 2,000 rows
gene$LoadFile(expression_file_name);

## Load covariates
cvrt = SlicedData$new();
cvrt$fileDelimiter = "\t";      # the TAB character
cvrt$fileOmitCharacters = "NA"; # denote missing values;
cvrt$fileSkipRows = 1;         # one row of column labels
cvrt$fileSkipColumns = 1;      # one column of row labels
if(length(covariates_file_name)>0) {
  cvrt$LoadFile(covariates_file_name);
}

# keep everything
filename = tempfile()
eqtls = Matrix_eQTL_main(
  snps = snps,
  gene = gene,
  cvrt = cvrt,
  output_file_name = "",
  pvOutputThreshold = 0,
  useModel = useModel,
  errorCovariance = numeric(),
  verbose = TRUE,
  output_file_name.cis = filename,
  pvOutputThreshold.cis = 1,
  snpspos = snpcoord,
  genepos = genecoords,
  cisDist = cisDist,
  pvalue.hist = "qqplot",
  min.pv.by.genesnp = FALSE,
  noFDRsaveMemory = FALSE);
unlink( filename )

```

We perform basic analysis to filter the data based on the nominal p-values then we save the data for further analysis. We keep all genes for which there is a p-value $< 10^{-4}$.

```
# initial analysis
pcut = 1e-4
eqtlmdat = eqtls$eqtls
sig.genes = eqtlmdat$gene[which(eqtlmdat$pvalue < pcut)] %>% unique %>% as.character
write.table(sig.genes,"ensID_1e-4_matrixeqtl.txt",sep="\t",quote=F,
  col.names=F,row.names=F)

# save data in .RData object
rm(snps,gene,cvrt,snpspos,genepos,expr_eqtl,expr1,expr0,resids_all,resids_pr)
save.image("eqtl_data.RData")
```

Here we perform comparison analyses between the eqtl results of 32 PEER factors, with the SVA covariates, and without the SVA covariates. First, perform the eqtl analysis with and without SVA results.

```
fname = "with_sva.txt"
sva0 <- read.table(fname,stringsAsFactors=F,header=T,sep="\t",check.names=F)

fname = "without_sva.txt"
nSva0 <- read.table(fname,stringsAsFactors=F,header=T,sep="\t",check.names=F)

fname = "snpcoordmat.txt"
snpcoord <- read.table(fname,stringsAsFactors=F,header=T,sep="\t",check.names=F)
fname = "genecoordmat.txt"
genecoords <- read.table(fname,stringsAsFactors=F,header=T,sep="\t",check.names=F)

# Performing the eqtl analysis with SVA result.

# eQTL analysis parameters/annotation
library(MatrixEQTL)
useModel = modelLINEAR_CROSS
SNP_file_name = "gtex_snp_mat.txt"
expression_file_name = "expr_eqtl.txt"
covariates_file_name = "with_sva.txt"
errorCovariance = numeric()
cisDist = 1e6

## Load genotype data
snps = SlicedData$new();
snps$fileDelimiter = "\t";      # the TAB character
snps$fileOmitCharacters = "NA"; # denote missing values;
snps$fileSkipRows = 1;         # one row of column labels
snps$fileSkipColumns = 1;      # one column of row labels
snps$fileSliceSize = 2000;     # read file in slices of 2,000 rows
snps$LoadFile(SNP_file_name);

## Load gene expression data
gene = SlicedData$new();
gene$fileDelimiter = "\t";      # the TAB character
gene$fileOmitCharacters = "NA"; # denote missing values;
gene$fileSkipRows = 1;         # one row of column labels
gene$fileSkipColumns = 1;      # one column of row labels
gene$fileSliceSize = 2000;     # read file in slices of 2,000 rows
gene$LoadFile(expression_file_name);

## Load covariates
```

```

cvrt = SlicedData$new();
cvrt$fileDelimiter = "\t";      # the TAB character
cvrt$fileOmitCharacters = "NA"; # denote missing values;
cvrt$fileSkipRows = 1;         # one row of column labels
cvrt$fileSkipColumns = 1;      # one column of row labels
if(length(covariates_file_name)>0) {
  cvrt$LoadFile(covariates_file_name);
}

# keep everything
filename = tempfile()
eqtls = Matrix_eQTL_main(
  snps = snps,
  gene = gene,
  cvrt = cvrt,
  output_file_name = "",
  pvOutputThreshold = 0,
  useModel = useModel,
  errorCovariance = numeric(),
  verbose = TRUE,
  output_file_name.cis = filename,
  pvOutputThreshold.cis = 1,
  snpspos = snpcoord,
  genepos = genecoords,
  cisDist = cisDist,
  pvalue.hist = "qqplot",
  min.pv.by.genesnp = FALSE,
  noFDRsaveMemory = FALSE);
unlink( filename )

eqtlmdat = eqtls$cis$eqtls

# initial analysis
rm(snps,gene,cvrt,snpspos,genepos,expr_eqtl,expr1,expr0,resids_all,resids_pr)
save.image("eqtl_data_S.RData")

# performing the SVA analysis without the SVA result (only the initial covariate).

# eQTL analysis parameters/annotation
library(MatrixEQTL)
useModel = modellINEAR_CROSS
SNP_file_name = "gtex_snp_mat.txt"
expression_file_name = "expr_eqtl.txt"
covariates_file_name = "without_sva.txt"
errorCovariance = numeric()
cisDist = 1e6

## Load genotype data
snps = SlicedData$new();
snps$fileDelimiter = "\t";      # the TAB character
snps$fileOmitCharacters = "NA"; # denote missing values;
snps$fileSkipRows = 1;         # one row of column labels
snps$fileSkipColumns = 1;      # one column of row labels
snps$fileSliceSize = 2000;     # read file in slices of 2,000 rows
snps$LoadFile(SNP_file_name);

## Load gene expression data
gene = SlicedData$new();

```

```

gene$fileDelimiter = "\t";      # the TAB character
gene$fileOmitCharacters = "NA"; # denote missing values;
gene$fileSkipRows = 1;         # one row of column labels
gene$fileSkipColumns = 1;      # one column of row labels
gene$fileSliceSize = 2000;     # read file in slices of 2,000 rows
gene$LoadFile(expression_file_name);

## Load covariates
cvrt = SlicedData$new();
cvrt$fileDelimiter = "\t";      # the TAB character
cvrt$fileOmitCharacters = "NA"; # denote missing values;
cvrt$fileSkipRows = 1;         # one row of column labels
cvrt$fileSkipColumns = 1;      # one column of row labels
if(length(covariates_file_name)>0) {
  cvrt$LoadFile(covariates_file_name);
}

# keep everything
filename = tempfile()
eqtls = Matrix_eQTL_main(
  snps = snps,
  gene = gene,
  cvrt = cvrt,
  output_file_name = "",
  pvOutputThreshold = 0,
  useModel = useModel,
  errorCovariance = numeric(),
  verbose = TRUE,
  output_file_name.cis = filename,
  pvOutputThreshold.cis = 1,
  snpspos = snpcoord,
  genepos = genecoords,
  cisDist = cisDist,
  pvalue.hist = "qqplot",
  min.pv.by.genesnp = FALSE,
  noFDRsaveMemory = FALSE);
unlink( filename )

eqtlmdat = eqtls$cis$eqtls

# initial analysis
rm(snps, gene, cvrt, snpspos, genepos, expr_eqtl, expr1, expr0, resids_all, resids_pr)
save.image("eqtl_data_NS.RData")

```

Compare the eqtl results from using 32 PEER factors, SVA results, and only the known covariates.

```

# load the eqtl results.

load("eqtl_data.RData")
eqtlmdat32 <- eqtlmdat

load("eqtl_data_NS.RData")
eqtlmdat = eqtls$cis$eqtls
eqtlmdatNS <- eqtlmdat

load("eqtl_data_S.RData")
eqtlmdat = eqtls$cis$eqtls
eqtlmdatS <- eqtlmdat

```

Here we set up a function to iteratively compare two eqtl results through a set of p value thresholds.

```
# set up a function for comparing the eqtl results.
compEqtl <- function(eq1, eq2, thr, wantData) {

  # declare and initialize the data table for the comparison.
  returnV <- c("pvalThreshold", "dat1", "dat2", "overlap",
    "overlapPercentage", "betaPearsonCorr", "betaSpearmanRankCorr",
    "pPearsonCorr", "pSpearmanRankCorr", "offDiagPercentage",
    "diagPercentage")
  returnData <- c()

  # loop through a set of p value thresholds that we are interested in
  for (th in thr) {
    # declare a list that we are storing the results for this iteration.
    loopV <- c()
    # add the threshold value to the list.
    loopV <- c(loopV, th)

    # filter the two data for our comparison with the p value threshold
    eq1F <- eq1[eq1$pvalue < th,]
    eq2F <- eq2[eq2$pvalue < th,]

    # sort the filtered data with respect to gene, pvalue, and snps.
    eq1Ordered <- eq1F[order(eq1F$gene, eq1F$pvalue, eq1F$snps),]
    eq2Ordered <- eq2F[order(eq2F$gene, eq2F$pvalue, eq2F$snps),]

    # create a list with unique gene symbols
    eq1FU <- unique(eq1F$gene)
    eq2FU <- unique(eq2F$gene)

    # find the index of rows with lowest p values and same associations.
    eq1UIndex <- match(eq1FU, eq1Ordered$gene)
    eq2UIndex <- match(eq2FU, eq2Ordered$gene)

    # extract the indexed rows from the filtered data.
    eq1U <- eq1Ordered[eq1UIndex,]
    eq2U <- eq2Ordered[eq2UIndex,]

    # find the length of the data after they are filtered.
    output1 <- length(eq1U$gene)
    output2 <- length(eq2U$gene)

    # add the number of rows from each data to the list.
    loopV <- c(loopV, output1)
    loopV <- c(loopV, output2)

    # make a list of associations for the filtered list.
    eq1A <- paste(eq1U$snps, eq1U$gene)
    eq2A <- paste(eq2U$snps, eq2U$gene)

    # find the list of associations that are shared between the two data.
    eq1M <- eq1A %in% eq2A
    eq2M <- eq2A %in% eq1A

    # select the rows and associations that are shared between the two data.
    eq1R <- eq1U[eq1M,]
```

```

eq2R <- eq2U[eq2M,]
eq1AR <- eq1A[eq1M]
eq2AR <- eq2A[eq2M]

# verify that the shared associations have identical length.
if(length(eq1AR) == length(eq2AR)) {
  print("No error in overlap")
} else {
  print("error in overlap")
}

# sort the data to have same order of associations.
eq1S <- eq1R
eq2S <- eq2R[match(eq1AR,eq2AR),]
output3 <- length(unique(eq1S$gene))
output33 <- length(unique(eq2S$gene))

# check whether the beta signs match for the shared data.
betaMatch <- sign(eq1S$beta) == sign(eq2S$beta)

# notify if the betas have same sign
if (all(betaMatch)) {
  print("beta dir same")
} else {
  print("beta dir diff")
}

# select the rows that have same beta sign and find the length.
eq1B <- eq1S[betaMatch,]
eq2B <- eq2S[betaMatch,]
output4 <- length(unique(eq1B$gene))
output44 <- length(unique(eq2B$gene))

# verify that the shared data have same length.
if(output4 == output44) {
  print("No error in beta")
} else {
  print("error in beta")
}

# add the number of shared data to the list.
loopV <- c(loopV, output4)

# find the percentage of the overlap with respect to the larger data.
largerN <- max(output1, output2)
output5 <- output4 / largerN * 100

# add the overlap percentage to the list.
loopV <- c(loopV, output5)

# find the total list of associations.
eq1I <- paste(eq1$snps, eq1$gene)
eq2I <- paste(eq2$snps, eq2$gene)

# find the list of shared unique associations.
eqA <- unique(c(eq1A, eq2A))

# find the index for shared unique associations in each data.

```

```

eq1UnI <- match(eqA, eq1I)
eq2UnI <- match(eqA, eq2I)

# find the union of the data under the threshold.
eq1Union <- eq1[eq1UnI,]
eq2Union <- eq2[eq2UnI,]

# verify that the union of data have same dimension.
if (all(eq1I[eq1UnI] == eq2I[eq2UnI])) {
  print("union ok")
} else {
  print("union error")
}

# check if the betas have same sign in each data.
betaAgree <- sign(eq1Union$beta) == sign(eq2Union$beta)

# notify if the betas have same directions.
if (all(betaAgree)) {
  print("beta dir same")
} else {
  print("beta dir diff")
}

# select the data with same beta sign within the union.
eq1UnionB <- eq1Union[betaAgree,]
eq2UnionB <- eq2Union[betaAgree,]

# find the percent proportion of the overlap with respect to the union.
output6 <- output4 / length(eq1Union$gene) * 100
output66 <- output4 / length(eq2Union$gene) * 100

# loopV <- c(loopV, output6)

# find the pearson correlation of the betas for the union.
output7 <- cor(eq1Union$beta, eq2Union$beta, method = "pearson")

loopV <- c(loopV, output7)

# find the spearman ranked correlation of the betas for the union.
output7_2 <- cor(eq1Union$beta, eq2Union$beta, method = "spearman")

loopV <- c(loopV, output7_2)

# find the pearson correlation of the -log10 p values for the union.
output7_3 <- cor(-log10(eq1Union$pvalue), -log10(eq2Union$pvalue),
  method = "pearson")

loopV <- c(loopV, output7_3)

# find the spearman ranked correlation of the -log10 p values for the union.
output7_4 <- cor(-log10(eq1Union$pvalue), -log10(eq2Union$pvalue),
  method = "spearman")

loopV <- c(loopV, output7_4)

# find the percent proportion of the data with different beta sign in union.
output8 <- ((length(eq1Union$gene) - length(eq1UnionB$gene)) /

```

```

    length(eq1Union$gene)) *100
output88 <- ((length(eq2Union$gene) - length(eq2UnionB$gene)) /
  length(eq2Union$gene)) * 100

# verify that they have same percent proportion.
if (output8 == output88) {
  print("diag ok")
} else {
  print("diag error")
}

loopV <- c(loopV, output8)

# find the percent proportion of the data with same beta sign in union.
output9 <- (length(eq1UnionB$gene) /
  length(eq1Union$gene)) *100

loopV <- c(loopV, output9)

# append the list of results the table of results.
returnV <- rbind(returnV, loopV)
if(wantData) {
  return(cbind(eq1UnionB,eq2UnionB$pvalue))
}
}
return(returnV)
}

```

Set up the list of thresholds and compare the eqtl results. Save the output as a table.

```

thresholds <- c(0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001, 0.0000001)

compNsS <- compEqtl(eqtldatNS, eqtldatS, thresholds, wantData=FALSE)
write.table(compNsS,"compNsS.txt",row.names=F,col.names=T,sep="\t",quote=F)

comp32Ns <- compEqtl(eqtldat32, eqtldatNS, thresholds, wantData=FALSE)
write.table(comp32Ns,"comp32Ns.txt",row.names=F,col.names=T,sep="\t",quote=F)

comp32S <- compEqtl(eqtldat32, eqtldatS, thresholds, wantData=FALSE)
write.table(comp32S,"comp32S.txt",row.names=F,col.names=T,sep="\t",quote=F)

save.image("comp_result.RData")

```

After consideration, we decided that $1e-4$ was a reasonable threshold to use.

```

thresholds <- c(1e-4)

compNsS <- compEqtl(eqtldatNS, eqtldatS, thresholds, wantData=TRUE)
write.table(compNsS,"compNsSData.txt",row.names=F,col.names=T,sep="\t",quote=F)

comp32Ns <- compEqtl(eqtldat32, eqtldatNS, thresholds, wantData=TRUE)
write.table(comp32Ns,"comp32NsData.txt",row.names=F,col.names=T,sep="\t",quote=F)

comp32S <- compEqtl(eqtldat32, eqtldatS, thresholds, wantData=TRUE)
write.table(comp32S,"comp32SData.txt",row.names=F,col.names=T,sep="\t",quote=F)

```


Identify the union data of the results. (Note that this was revised for the manuscript).

```
compNsS <- read.table("compNsSData.txt",header=T,sep="\t",stringsAsFactors=F)
comp32Ns <- read.table("comp32NsData.txt",header=T,sep="\t",stringsAsFactors=F)
comp32S <- read.table("comp32SData.txt",header=T,sep="\t",stringsAsFactors=F)

fitlerPval <- 1e-3

comp32SP <- comp32S[comp32S$pvalue <= fitlerPval &
  comp32S$"eq2UnionB$pvalue" <= fitlerPval,]
compNsSP <- compNsS[compNsS$pvalue <= fitlerPval &
  compNsS$"eq2UnionB$pvalue" <= fitlerPval,]

comp32NsP <- comp32Ns[comp32Ns$pvalue <= fitlerPval &
  comp32Ns$"eq2UnionB$pvalue" <= fitlerPval,]

comp32ScompNsS <- rbind(comp32S,compNsS)

comp32Nscomp32S <- rbind(comp32Ns, comp32S)

unionEqtl <- function(eq) {
  associ <- paste(eq$snps,eq$gene,sep="")
  associUni <- unique(associ)
  uniIndex <- match(associUni, associ)
  outputData <- eq[uniIndex,]
  return(outputData)
}

comp32ScompNsSUni <- unionEqtl(comp32ScompNsS)
comp32Nscomp32SUni <- unionEqtl(comp32Nscomp32S)

comp1Asso <- paste(comp32ScompNsSUni$snps, comp32ScompNsSUni$gene,sep="")
comp2Asso <- paste(comp32Nscomp32SUni$snps, comp32Nscomp32SUni$gene,sep="")

check1in2 <- comp1Asso %in% comp2Asso
check2in1 <- comp2Asso %in% comp1Asso

all(check1in2)
all(check2in1)

intersect1 <- comp32ScompNsSUni[check1in2,]
intersect2 <- comp32Nscomp32SUni[check2in1,]

intersect1Asso <- paste(intersect1$snps, intersect1$gene,sep="")
intersect2Asso <- paste(intersect2$snps, intersect2$gene,sep="")

check1in2 <- intersect1Asso %in% intersect2Asso
check2in1 <- intersect2Asso %in% intersect1Asso

all(check1in2)
all(check2in1)

write.table(intersect1,"intersect1.txt",row.names=F,col.names=T,sep="\t",quote=F)
```

Output the chromosome number, chromosome position, reference allele, alternative allele, gene symbol and pvalues as a table.

```
associations <- intersect1[,1:2]

ensToGeneNInd <- match(associations$gene, ann_gene0$gene_id)
ensToGeneN <- ann_gene0$gene_name[ensToGeneNInd]

splt = sapply(associations$snps,function(x){
  out1 = strsplit(x,"_")[[1]]
  return(out1)
})

chr <- splt[1,]
pos <- splt[2,]

ref <- splt[3,]
alt <- splt[4,]
gene <- ensToGeneN

pval <- intersect1[,4]

outAsso <- as.data.frame(cbind(chr, pos, ref, alt, gene, pval))

write.table(outAsso,"associations.txt",row.names=F,col.names=T,sep="\t",quote=F)
```