Efficient and Effective Control of Confounding in eQTL Mapping Studies through Joint Differential Expression and Mendelian Randomization Analyses–analysis script

Yue Fan and Xiang Zhou

2020-06-02

Introduction

This vignette provides the analysis scripts for GTEx data in details.

Real data processing

… implys the workdir

Extract the information of genes (Require R package data.table)

```
cd .../expression

for FILE in *.bed.gz
do
echo ${FILE}
ARR=($(echo ${FILE} | sed 's/.v7.normalized_expression.bed.gz/\n/g'))  ##GTEx gene expressoin
file
echo ${ARR}
zcat ${FILE} | cut -f1-4 > .../expression/${ARR}_gene_info.txt
done

for FILE in *_gene_info.txt
do
echo ${FILE}
ARR=($(echo ${FILE} | sed 's/_gene_info.txt/\n/g'))
echo ${ARR}
Rscript cis.R ${ARR}
done

###cis.R####
args <- commandArgs(TRUE)
tissue <- as.character(args[1]) ##the tissue's name(Muscle Skeletal,...)

library(data.table)
gene_info <- data.frame(fread(paste(tissue, "_gene_info.txt", sep=""), header=T))
for (i in 1:nrow(gene_info)) {
  gene_info[i,2] <- max(0, gene_info[i,2]-10^6)
  gene_info[i,3] <- gene_info[i,3]+10^6
}
write.table(gene_info, paste(tissue, "_gene_info1.bed", sep=""), quote=F, col.names=F,
row.names=F)
```

Extract the phenotype data (Height, weight and BMI)

cd …/GTEx/64278/PhenoGenotypeFiles/RootStudyConsentSet_phs000424.GTEx.v7.p2.c1.GRU/PhenotypeFiles gunzip -c phs000424.v7.pht002742.v7.p2.c1.GTEx_Subject_Phenotypes.GRU.txt.gz > …/phenotype/phenotype.txt

cat phenotype.txt| awk '{print $1" "$2" "$4" "$5" "$6" "$7" "$8" "$10" "$12" "$13" "$14" "$15" "$16" "$17}' > sub_phenotype.txt

clean sub_phenotype.txt dt <- read.table("sub_phenotype.txt", header=T) for (i in 1:nrow(dt)) { if (dt$SEX[i]=="Donor") { print (i) dt[i,3:5] <- dt[i,5:7] dt[i,7:8] <- dt[i,8:9] dt[i,9] <- dt[i,11] } }

dt <- dt[,1:9]; dt <- dt[,-6] write.table(dt, "sub_phenotype1.txt", quote=F, col.names=T, row.names=F)

Extract the genotype data for each gene (Require intersectBed)

```
for ((i=1; i<=22; i++)); do
echo ${i}
VCFFILE=.../genotype/imputed/data/chr${i}.dose.vcf.gz
zcat ${VCFFILE} | cut -f1-8 > chr${i}.vcf
cat chr${i}.vcf | awk -F ';' '{print $1"\t"$2"\t"$3 }' > chr${i}.bed
done

for ((i=1; i<=22; i++)); do
echo ${i}
awk '{sub("MAF=", "", $9); print}' < chr${i}.bed > chr_${i}.bed
done

for ((i=1; i<=22; i++)); do
echo ${i}
awk '{ if(($9 >= 0.05)) { print } }' chr_${i}.bed > qc/chr${i}.bed
done

for FILE in *_gene_info1.bed
do
echo ${FILE}
ARR=($(echo ${FILE} | sed 's/_gene_info1.bed/\n/g'))
echo ${ARR}
awk  'BEGIN{ OFS="\t"; }{ print $1, $2, $3, $4; }' ${FILE} > ${ARR}_gene_info.bed
done

cd .../qc
GENE=...
INTERSECTBED=.../bedtools/bedtools2/bin/intersectBed
cd ${GENE}
for FILE in `ls *_gene_info.bed`; do
echo ${FILE}
ARR=($(echo ${FILE} | sed 's/_gene_info.bed/\n/g')); echo ${ARR}
mkdir .../qc/${ARR}
 cd /net/mulan/disk2/yuef/data/GTEX/GTEx_v7/qc
for ((i=1; i<=22; i++)); do
${INTERSECTBED} -a all_chr${i}.bed  -b ${GENE}/${FILE} -wb > ./${ARR}/${ARR}_chr${i}
```

```
cd ./${ARR}
awk '{print >> $645; close($645)}' ${ARR}_chr${i}
done
done

library(doParallel) ###require R package DoParallel, data.table

args <- commandArgs(TRUE)
args <- as.numeric(args)
chr=args[1]  ###chromosome
d=read.table('gene.bed')
library(data.table)
library(doParallel)
numCore = 10     ### numCore is the number of cores in your CPU

idx=which(d[,1]==chr)
d=d[idx,]

registerDoParallel(cores=numCore)
N=nrow(d)
resBMM <- foreach(i=1:N, .combine=rbind, .errorhandling = 'remove')%dopar%
{
  res <- data.frame()
name=d[i,4]
name=as.character(name)
file= (‘#########’) ###load the genotype data of the cis-SNP of ith gene
if(file.exists(file))
{
  snp_raw=fread(file)
  snp_raw=snp_raw[,7:641]
  sfile=paste0('###') ###save the processed genotype data
  save(snp_raw,file=sfile)
}
print(i)
res <- data.frame(iter=i)
return(res)
}
```

Calculate the gene expression residuals with PEER pacakge(Require R package peer)

```
cd .../expression/
for ((i=1;i<=48;i++))
do
for j in 1 2 5 10 15 20 25 30 35 40 50 60 70 80 90 100 150 200 250
do
Rscript calc_peer.R ${i} ${j}
done
done

###calc_peer.R
args <- commandArgs(TRUE)
tissue <- as.numeric(args[1])
pc<-as.numeric(args[2])
library('data.table')
tissue_info <- fread("######") ###load a file contains the names of 48 tissues
```

```
tissue=tissue_info[tissue]

library(data.table)
library(peer)
dt <- data.frame(fread(paste(tissue, "_expression.txt", sep=""), header = T))
dt <- dt[,-c(1:4)]; dt <- t(as.matrix(dt))
  model = PEER()
  PEER_setPhenoMean(model,as.matrix(dt))
  dim(PEER_getPhenoMean(model))

  PEER_setAdd_mean(model, TRUE)
  PEER_setNk(model,pc)  ## pc hidden confounders
  PEER_getNk(model)

  PEER_update(model)
  #PEER_setNMax_iterations(model, 10000)

  factors = PEER_getX(model)
  factors=factors[,-1]
  #dim(factors)
  residuals = PEER_getResiduals(model)
  #dim(residuals)
  #write.table(factors, paste("peer_factor_",tissue,'_', pc, ".txt", sep=""), quote=F,
col.names=F, row.names=F)
write.table(residuals, paste(tissue,'_peer', pc, ".txt", sep=""), quote=F, col.names=F,
row.names=F)
  print (pc)
```

Generate Simulation data (Require R package data.table)

```
library(data.table)
n = 491 ###sample size
n_genes = 10000 ### number of genes
n_conf = 10 # number of confounding effects
pve1 = 0.03; # the genetic contribution to the gene expression variation
pve2 = 0.5;  # the  confounding factors in total contribute to the gene expression variation
pve3 = 0.25; # all genes in total contribution to the phenotypic variance
C <- matrix(rnorm(n*n_conf, 0, sd = 1), nrow=n) # confounding factors
M_matrix <- c(); Y_matrix <- c()
g_matrix <- c(); beta_M1_matrix <- c()

gene <- fread(paste("/net/mulan/disk2/yuef/data/GTEX/GTEx_v7/expression/", tissue,
"_expression.txt", sep=""), header=T) #gene expression data
gene_id <- colnames(gene)[-c(1:4)]
gene_names <- gene$gene_id
common <- gene_id
num=length(gene_names)
idx=sample(1:num) # we select 10000 genes randomly
num=1:n_genes
gene_names=gene_names[num]
```

We first generate the gene expression data

```
for(i in 1:n_genes)
{
  #idx=sample(1:10)[1:5] #heterogeneous confounding scenario
  file=paste(".../qc/", tissue, "/", gene_names[i], sep="")
  snp_raw <- fread(file)
  snp_raw <- data.frame(snp_raw[,7:641])
  A <- t(snp_raw[,geno_id %in% common])
  num_snp=ncol(A)


  j=sample(1:num_snp,1)
  g1 <- A[,j]

  beta_g1 <- rnorm(1, 0, sd = 1)
  beta_g1 <- beta_g1/(sd(g1 * beta_g1)) * sqrt(pve1)

  beta_C <- rnorm(n_conf, 0, sd = 1)
  beta_C <- beta_C/(sd(C %*% beta_C)) * sqrt(pve2)
  #  beta_C[idx] <- beta_C[idx]/(sd(C[,idx] %*% beta_C[idx])) * sqrt(pve2)# heterogeneous
confounding scenario
  e1 <- rnorm(n, 0, sd = sqrt(1-pve1-pve2))

  g_matrix <- cbind(g_matrix, g1 * beta_g1 + e1)
  M_matrix <- cbind(M_matrix, g1 * beta_g1 + C %*% beta_C + e1)
 # M_matrix <- cbind(M_matrix, g1 * beta_g1 + C[,idx] %*% beta_C[idx] + e1)# heterogeneous
confounding scenario
}
```

Generating the phenotype data

```
gamma <- array(0, n_genes)
gamma[1:1000] <- rnorm(1000, 0, 1)
gamma <- gamma/(sd(g_matrix %*% gamma)) * sqrt(pve_M)
e2 <- rnorm(n, 0, sd = sqrt(1-pve_M))
Y <- g_matrix %*% gamma + e2
save(pc_matrix,Y,A,M_matrix,file='sim_pc10_sparse.RData') #sparse setting

gamma <- array(0, n_genes)
gamma[1:10000] <- rnorm(10000, 0, 1)
gamma <- gamma/(sd(g_matrix %*% gamma)) * sqrt(pve_M)
e2 <- rnorm(n, 0, sd = sqrt(1-pve_M))
Y <- g_matrix %*% gamma + e2
save(pc_matrix,Y,A,M_matrix,file='sim_pc10_poly.RData') # polygenic setting
```

Select the instrumental variable for each gene (Require R package MatrixEQTL)

```
library(MatrixEQTL)
###################
### Read files ####
###################
iv_snp=c()
peer=0
  for(iter in iteration)
  {

    M <- M_matrix[,iter]


    load('#########') ###load the genotype data of the cis-SNP of iterth gene
    snp_raw <- data.frame(snp_raw)
    A <- t(snp_raw[,geno_id %in% common])


    snps = SlicedData$new();
    A=as.matrix(A)
    A=t(A)
    snps$CreateFromMatrix(A)     #Load genotype data

    gene=SlicedData$new();
    res1=as.matrix(M)
    res1=t(res1)
    gene$CreateFromMatrix(res1)  #Load gene expression data

    useModel = modelLINEAR;
    pvOutputThreshold = 5e-1;
    errorCovariance = numeric();
    output_file_name = tempfile();
    me = Matrix_eQTL_engine(         # Run the analysis
      snps = snps,
      gene = gene,
      output_file_name = output_file_name,
      pvOutputThreshold = pvOutputThreshold,
      useModel = useModel,
      errorCovariance = errorCovariance,
      verbose = TRUE,
      pvalue.hist = TRUE,
      min.pv.by.genesnp = FALSE,
      noFDRsaveMemory = FALSE);
    time=as.numeric(me$time.in.sec)
    me=me$all$eqtls[1,]
    tmp=as.character(me$snps)
    onesnp=as.numeric(substr(tmp,4,nchar(tmp)))

    iv_snp=rbind(iv_snp,A[onesnp,])           ##save the instrumental variable for each gene
    num_snp=rbind(num_snp,c(iter,num1,num2))


    l1 <- lm(M ~ A[onesnp,])
    l2 <- lm(Y ~ M)
    l3 <- lm(Y ~  A[onesnp,])


    summary <- rbind(summary, c(iter,summary(l1)$coeff[2,4], summary(l1)$coeff[2,1],
summary(l3)$coeff[2,1],summary(l3)$coeff[2,4], summary(l3)$coeff[2,1]/summary(l1)$coeff[2,1],
summary(l2)$coeff[2,1],summary(l2)$coeff[2,4],time))

    print(iter)
  }

  write.table(summary, paste("mr2_sim_", pc, ".txt", sep=""), quote=F, col.names=F, row.names=F)
  file=paste("mr_sim.RData", sep="")
  save(iv_snp,file=file)
```

```
args <- commandArgs(TRUE)
peer <- as.integer(args[1])
###################
### Read files ####
###################
  for(iter in 1:N)
  {
    ind=num_snp[iter,1]
    res1=gene_peer[,ind]
    M <- M_matrix[,ind]
    A=iv_snp[iter,]


    t1=Sys.time()
    l1 <- lm(M ~ A)

    l2 <- lm(Y ~ res1)

    l3 <- lm(Y ~ A)
    liv=ivreg(Y~M,~A)
    t2=Sys.time()
    time=as.numeric(t2-t1)

    summary <- rbind(summary, c(iter,summary(l1)$coeff[2,4], summary(l1)$coeff[2,1],
summary(l3)$coeff[2,1],summary(l3)$coeff[2,4],
summary(l3)$coeff[2,1]/summary(l1)$coeff[2,1],summary(liv)$coeff[2,4],
summary(l2)$coeff[2,1],summary(l2)$coeff[2,4],time))


  }

  write.table(summary, paste("mr2_sim_peer", peer, ".txt", sep=""), quote=F, col.names=F,
row.names=F)
```