

Sex-based differential gene expression analysis

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	Differential expression analysis for GTEx subcutaneous adipose tissue	2
3	Differential expression analysis for deCODE subcutaneous adipose tissue	5
4	Differential expression analysis for AAGMEx subcutaneous adipose tissue	7
5	Combining differential expression analyses	8
6	References	13

List of Figures

1	Differential expression analysis from GTEx (red, higher in females; blue, higher in makes).	4
2	Differential expression analysis from deCODE (red, higher in females; blue, higher in makes).	6
3	Differential expression analysis from AAGMEx (red, higher in females; blue, higher in makes).	8
4	Venn-diagram delineating the differentially expressed genes in the three different data groups.	12

1 Overview

The following analyses document our methods for differential gene expression analysis.

2 Differential expression analysis for GTEx subcutaneous adipose tissue

We analyzed expression data that were corrected for known and unknown covariates (see *FigS3.GTEx.pdf*). First we import the data and implement basic processing.

```
library(dplyr)
library(limma)

# import data
fname = "gencode_gene_map.txt"
ann_gene0 = read.table(fname,header=T,sep="\t",stringsAsFactors=F,quote = "")
fname = "genes_XYM.txt"
genes_XYM = read.table(fname,header=F,sep="\t",stringsAsFactors=F,quote = "")
fname = "gtex_subq_covars.txt"
demo0 = read.table(fname,header=T,sep="\t",stringsAsFactors=F,quote = "")
fname = "gtex_subq_expr_sva.txt"
expr0 = read.table(fname,header=T,sep="\t",stringsAsFactors=F,check.names=F,quote = "")

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

# get gene annotation for expressed transcripts
inds0 = match(rownames(expr0), ann_gene0$gene_id)
inds = inds0[!is.na(inds0)]
gene.map = (ann_gene0[inds,])
all(gene.map$gene_id == rownames(expr0))

# omit transcripts on sex chromosomes
ind.sex0 = match(genes_XYM[,1], gene.map$gene_name)
ind.sex = ind.sex0[!is.na(ind.sex0)]
expr1 = expr0[-ind.sex,]
gene.map = gene.map[-ind.sex,]

# specify groups for comparison
ind_F = which(demo0$GENDER == "Female")
ind_M = which(demo0$GENDER == "Male")
null <- "Female"
alt <- "Male"
expr_dat = expr1
dat_Null = t(expr_dat[,ind_F]) # Null model - female
dat_Alt = t(expr_dat[,ind_M]) # Alt model - male
```

Next we establish a function that will be used for all sex-based differential expression analyses. We also specify a plotting function for an 'MA plot' in which the log₂ fold change is plotted with respect to the average expression. We annotate and operationally define differentially expressed genes (DEGs) as those with $|\text{fold change}| > 1.05$ and $\text{FDR} < 0.05$.

```
# function for DEG analysis
deg.analysis = function(dat_Null=NULL,dat_Alt=NULL,null=NULL,alt=NULL){

  # implement linear model analysis with eBayes and BH adjustments
  subQall = rbind(dat_Null, dat_Alt)
  subQsex = c(rep(null,nrow(dat_Null)), rep(alt,nrow(dat_Alt)))
  design <- model.matrix(~0+subQsex)
  colnames(design) <- c(null,alt)
  contrast <- makeContrasts(Female - Male, levels = design)
  fit <- lmFit(t(subQall), design)
  fit <- contrasts.fit(fit, contrast) %>% eBayes
  output0 <- topTable(fit,number=ncol(subQall),adjust.method="BH")

  # convert log2FC to foldchange increase/decrease
  output = output0 %>% mutate(FC = 2^logFC)
  fc = rep(1,nrow(output))
  for (ii in 1:nrow(output)){
    if(output$FC[ii] > 1){fc[ii] = output$FC[ii]}
    if(output$FC[ii] < 1){fc[ii] = -1/output$FC[ii]}
  }
  output = output %>% mutate(ratioFC = fc) %>% mutate(absFC = abs(ratioFC))
  rownames(output) = rownames(output0)

  return(output)
}

# function for plotting DEG analysis results
deg.ma.plot = function(data=NULL,fname=NULL,fc.cut=1.05,fdr.cut=0.05){
  pdf(fname,height=4,width=4)
  outf = output %>% filter(logFC > log2(fc.cut), adj.P.Val < fdr.cut)
  outm = output %>% filter(logFC < -log2(fc.cut), adj.P.Val < fdr.cut)
  plot(output$AveExpr, output$logFC, col="gray",
        xlab="Average expression",ylab="Log2 fold change")
  points(outf$AveExpr, outf$logFC, col="red")
  points(outm$AveExpr, outm$logFC, col="blue")
  abline(h=0,lty=2)
  dev.off()
}
```

We then implement the differential expression analysis, plot the results and write the output to file.

```
# implement DEG analysis and plot results
output = deg.analysis(dat_Null=dat_Null,dat_Alt=dat_Alt,null=null,alt=alt)
deg.ma.plot(data=output,fname="gtex_maplot.pdf")

# add gene names and write the data to file
gene.inds = match(rownames(output),gene.map$gene_id)
output = output %>% mutate(gene = gene.map$gene_name[gene.inds])
write.table(output,"gtex_deg.txt",col.names=T,row.names=F,sep="\t",quote=F)

# for subsequent plotting
dat_F_gtex = dat_Null
dat_M_gtex = dat_Alt
```

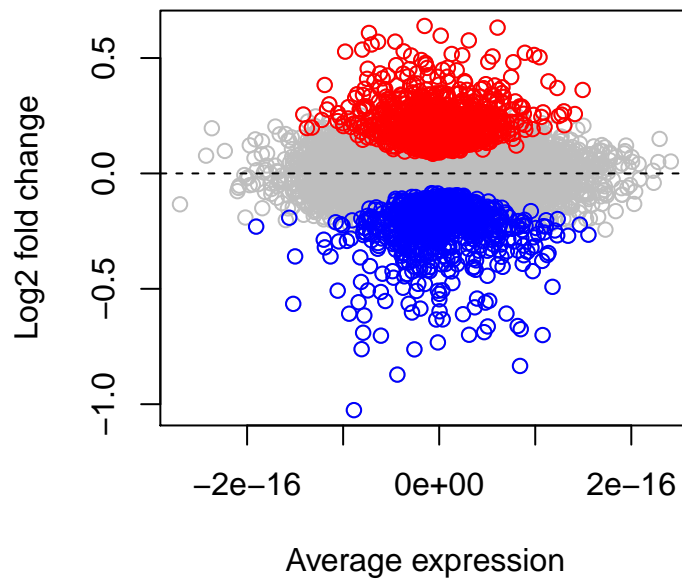


Figure 1: Differential expression analysis from GTEx (red, higher in females; blue, higher in males).

The results are shown in Figure 1. Genes with elevated expression in females are indicated by red, while genes with elevated expression in males are indicated by blue.

3 Differential expression analysis for deCODE subcutaneous adipose tissue

We analyzed expression data that were downloaded from GEO and annotated as described in (see *Fig1.GEOdata.pdf*). We implement the differential expression analysis as described above. Note that the above functions need to be run, *deg.analysis* and *deg.ma.plot*.

```
library(dplyr)
library(limma)

# import data
fname = "expr_data_decode_qc.txt"
expr0 = read.table(fname,header=T,sep="\t",stringsAsFactors=F)
fname = "genes_decode.txt"
ann_gene0 = read.table(fname,header=T,sep="\t",stringsAsFactors=F,quote="")
fname = "genes_XYM.txt"
genes_XYM = read.table(fname,header=F,sep="\t",stringsAsFactors=F)
fname = "phenotypes_decode_qc.txt"
demo0 = read.table(fname,header=T,sep="\t",stringsAsFactors=F)

# check data organization
all(demo0$sample == names(expr0))

# get gene annotation for expressed transcripts
inds = match(rownames(expr0),ann_gene0$ID)
gene.map = ann_gene0[inds,]
all(gene.map$ID == rownames(expr0))

# omit transcripts on sex chromosomes
ind.sex0 = match(genes_XYM[,1],gene.map$gene_list)
ind.sex = ind.sex0[!is.na(ind.sex0)]
expr1 = expr0[-ind.sex,]
gene.map = gene.map[-ind.sex,]

# specify groups for comparison
ind_F = which(demo0$sex == "Female")
ind_M = which(demo0$sex == "Male")
null <- "Female"
alt <- "Male"
expr_dat = expr1
dat_Null = t(expr_dat[,ind_F]) # Null model - female
dat_Alt = t(expr_dat[,ind_M]) # Alt model - male

# implement DEG analysis and plot results
output = deg.analysis(dat_Null=dat_Null,dat_Alt=dat_Alt,null=null,alt=alt)
deg.ma.plot(data=output,fname="decode_maplot.pdf")

# write the data to file
gene.inds = match(rownames(output),gene.map$ID)
output = output %>% mutate(gene = gene.map$gene_list[gene.inds])
write.table(output,"decode_deg.txt",col.names=T,row.names=F,sep="\t",quote=F)

# for subsequent plotting
dat_F_decode = dat_Null
dat_M_decode = dat_Alt
```

The result is plotted in Figure 2.

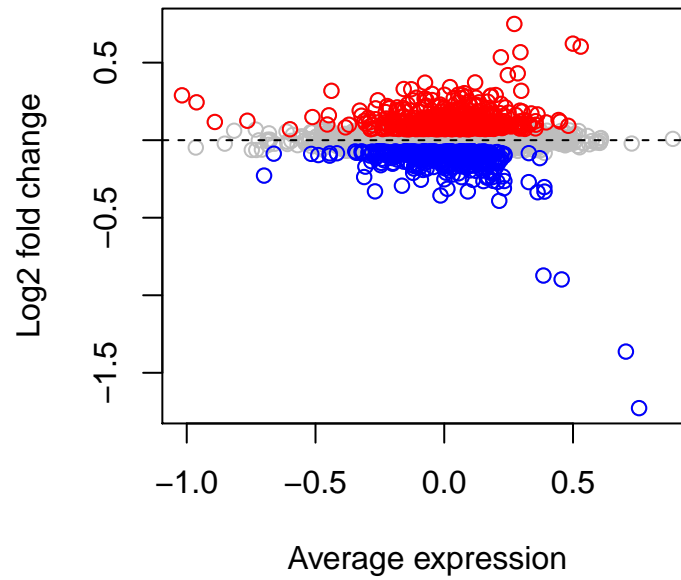


Figure 2: Differential expression analysis from deCODE (red, higher in females; blue, higher in males).

4 Differential expression analysis for AAGMEx subcutaneous adipose tissue

We analyzed expression data that were downloaded from GEO and annotated as described in (see *Fig1.GEOdata.pdf*). We implement the differential expression analysis as described above. Note that the above functions need to be run, *deg.analysis* and *deg.ma.plot*.

```
library(dplyr)
library(limma)

# import data
fname = "expr_data_aagmex_qc.txt"
expr0 = read.table(fname,header=T,sep="\t",stringsAsFactors=F)
fname = "genes_XYM.txt"
genes_XYM = read.table(fname,header=F,sep="\t",stringsAsFactors=F)
fname = "phenotypes_aagmex_qc.txt"
demo0 = read.table(fname,header=T,sep="\t",stringsAsFactors=F)

# check data organization
all(demo0$accession == names(expr0))

# omit transcripts on sex chromosomes
ind.sex0 = match(genes_XYM[,1], rownames(expr0))
ind.sex = ind.sex0[!is.na(ind.sex0)]
expr1 = expr0[-ind.sex,]

# specify groups for comparison
ind_F = which(demo0$sex == "Female")
ind_M = which(demo0$sex == "Male")
null <- "Female"
alt <- "Male"
expr_dat = expr1
dat_Null = t(expr_dat[,ind_F]) # Null model - female
dat_Alt = t(expr_dat[,ind_M]) # Alt model - male

# implement DEG analysis and plot results
output = deg.analysis(dat_Null=dat_Null,dat_Alt=dat_Alt,null=null,alt=alt)
deg.ma.plot(data=output,fname="aagmex_maplot.pdf")

# write the data to file
output = output %>% mutate(gene = rownames(output))
write.table(output,"aagmex_deg.txt",col.names=T,row.names=F,sep="\t",quote=F)

# for subsequent plotting
dat_F_aagmex = dat_Null
dat_M_aagmex = dat_Alt
```

The result is illustrated in Figure 3.

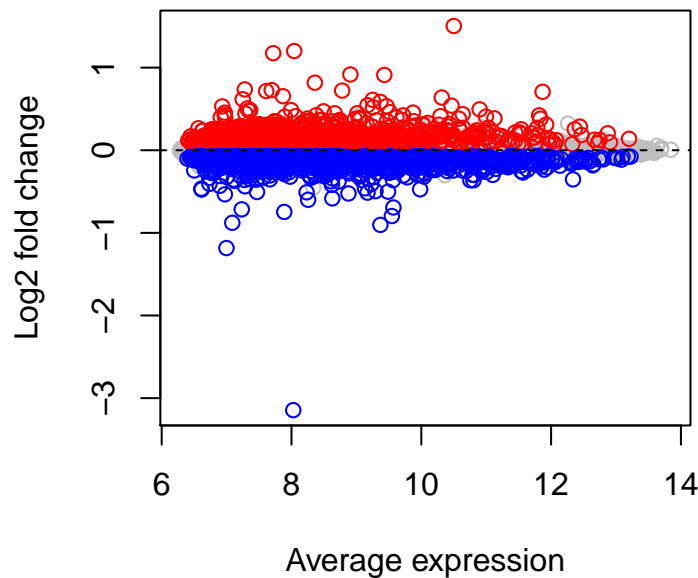


Figure 3: Differential expression analysis from AAGMEx (red, higher in females; blue, higher in males).

5 Combining differential expression analyses

Here we aggregate the data from the differential expression analyses of individual data sets to determine which genes show sign-consistent differential expression across the three data sets. For GTEx and deCODE, multiple transcripts/probe sets were quantified for individual genes. In such cases, we considered only the entry with the minimal adjusted p-value for a given gene.

```
library(dplyr)

# import subq data:
gtex0 = read.table("gtex_deg.txt", sep="\t", stringsAsFactors=F, header=T, quote="")
decode0 = read.table("decode_deg.txt", sep="\t", stringsAsFactors=F, header=T, quote="")
aagmex0 = read.table("aagmex_deg.txt", sep="\t", stringsAsFactors=F, header=T, quote="")
```

Filter the data with the minimal p-value.

```
# look at uniqueness of gene IDs
nrow(gtex0) == length(unique(gtex0$gene)) # FALSE
nrow(decode0) == length(unique(decode0$gene)) # FALSE
nrow(aagmex0) == length(unique(aagmex0$gene)) # TRUE

# function for getting indices of unique genes with min p-values
min.p.indices = function(dat=NULL){
  ind_keep = c()
  for(ii in 1:length(unique(dat$gene))){
    ind = which(dat$gene == unique(dat$gene)[ii])
    if(length(ind)==1){ind_keep = c(ind_keep, ind)}
    if(length(ind)>1){
```



```

        ind2 = which(dat$adj.P.Val[ind] == min(dat$adj.P.Val[ind]))
        ind_keep = c(ind_keep, ind[ind2])
    }
}
if(length(ind_keep)==length(unique(dat$gene))){return(ind_keep)}
}

# get the data sets with unique genes
decode1 = decode0[min.p.indices(decode0),]
gtex1 = gtex0[min.p.indices(gtex0),]

```

Find the overlap of DEGs between gtex and decode data.

```

# get overlapping set
overlap_genes = intersect(gtex1$gene, decode1$gene) %>% unique

# get indices of overlapping genes
gtex_inds0 = match(overlap_genes, gtex1$gene)
gtex_inds = gtex_inds0[!is.na(gtex_inds0)]
decode_inds0 = match(overlap_genes, decode1$gene)
decode_inds = decode_inds0[!is.na(decode_inds0)]
all.equal(gtex1$gene[gtex_inds], decode1$gene[decode_inds])

# specify new data
gtex_decode = cbind(gtex1$gene[gtex_inds],
  gtex1[gtex_inds,c(8,5)], decode1[decode_inds,c(8,5)])
names(gtex_decode) = c("gene","gtex_fc","gtex_fdr","decode_fc","decode_fdr")

# identify matching foldchange at FDR < 0.05
gtex_decode_fc = gtex_decode %>% mutate(fc_prod = sign(gtex_fc*decode_fc)) %>%
  filter(fc_prod==1,gtex_fdr<0.05,decode_fdr<0.05) %>%
  filter(abs(gtex_fc)>1.05,abs(decode_fc)>1.05)

```

Find the overlap of DEGs between gtex and aagmex data.

```

# get overlapping set
overlap_genes = intersect(gtex1$gene, aagmex0$gene) %>% unique

# get indices of overlapping genes
gtex_inds0 = match(overlap_genes, gtex1$gene)
gtex_inds = gtex_inds0 [!is.na(gtex_inds0 )]
aagmex_inds0 = match(overlap_genes, aagmex0$gene)
aagmex_inds = aagmex_inds0[!is.na(aagmex_inds0)]
all.equal(gtex1$gene[gtex_inds], aagmex0$gene[aagmex_inds])

# specify new data
gtex_aagmex = cbind(gtex1$gene[gtex_inds],
  gtex1[gtex_inds,c(8,5)], aagmex0[aagmex_inds,c(8,5)])
names(gtex_aagmex) = c("gene","gtex_fc","gtex_fdr","aagmex_fc","aagmex_fdr")

# identify matching foldchange at FDR < 0.05
gtex_aagmex_fc = gtex_aagmex %>% mutate(fc_prod = sign(gtex_fc*aagmex_fc)) %>%
  filter(fc_prod==1,gtex_fdr<0.05,aagmex_fdr<0.05) %>%
  filter(abs(gtex_fc)>1.05,abs(aagmex_fc)>1.05)

```

Find the overlap of DEGs between aagmex and decode data.

```
# get overlapping set
overlap_genes = intersect(decode1$gene, aagmex0$gene) %>% unique

# get indices of overlapping genes
decode_inds0 = match(overlap_genes, decode1$gene)
decode_inds = decode_inds0[!is.na(decode_inds0)]
aagmex_inds0 = match(overlap_genes, aagmex0$gene)
aagmex_inds = aagmex_inds0[!is.na(aagmex_inds0)]
all.equal(decode1$gene[decode_inds], aagmex0$gene[aagmex_inds])

# specify new data
decode_aagmex = cbind(decode1$gene[decode_inds],
  decode1[decode_inds,c(8,5)], aagmex0[aagmex_inds,c(8,5)])
names(decode_aagmex) = c("gene","decode_fc","decode_fdr","aagmex_fc","aagmex_fdr")

# identify matching foldchange at FDR < 0.05
decode_aagmex_fc = decode_aagmex %>% mutate(fc_prod = sign(decode_fc*aagmex_fc)) %>%
  filter(fc_prod==1,decode_fdr<0.05,aagmex_fdr<0.05) %>%
  filter(abs(decode_fc)>1.05,abs(aagmex_fc)>1.05)
```

Find the overlap of DEGs among all three data sets.

```
# get overlapping set
overlap_genes = Reduce(intersect, list(gtex1$gene, decode1$gene, aagmex0$gene)) %>%
  unique

# get indices
gtex_inds0 = match(overlap_genes, gtex1$gene)
gtex_inds = gtex_inds0[!is.na(gtex_inds0)]
decode_inds0 = match(overlap_genes, decode1$gene)
decode_inds = decode_inds0[!is.na(decode_inds0)]
aagmex_inds0 = match(overlap_genes, aagmex0$gene)
aagmex_inds = aagmex_inds0[!is.na(aagmex_inds0)]
all.equal(gtex1$gene[gtex_inds], decode1$gene[decode_inds], aagmex0$gene[aagmex_inds])

# specify new data
all = cbind(gtex1$gene[gtex_inds],
  gtex1[gtex_inds,c(8,5)], decode1[decode_inds,c(8,5)], aagmex0[aagmex_inds,c(8,5)],
  stringsAsFactors=F)
names(all) = c("gene","gtex_fc","gtex_fdr","decode_fc",
  "decode_fdr","aagmex_fc","aagmex_fdr")

# isolate data for genes with identical sign fold changes and FDR < 0.05
ind_keep = c()
for(ii in 1:nrow(all)){
  a = sign(all$gtex_fc[ii])
  b = sign(all$decode_fc[ii])
  c = sign(all$aagmex_fc[ii])
  if(all.equal(a,b,c)==T){ind_keep = c(ind_keep,ii)}
}
all_fc = all[ind_keep,] %>% filter(gtex_fdr<0.05, decode_fdr<0.05, aagmex_fdr<0.05) %>%
  filter(abs(gtex_fc)>1.05, abs(decode_fc)>1.05, abs(aagmex_fc)>1.05)

write.table(all_fc,"all_fc_v8.txt",col.names=T,row.names=F,sep="\t",quote=F)
```

Using the overlap of DEGs, create a venn diagram that illustrates the number of DEGs in each data set and the overlap of the DEGs among them.

```
library(VennDiagram)

gtex = nrow(gtex1 %>% filter(adj.P.Val<0.05,abs(ratioFC)>1.05))
decode = nrow(decode1 %>% filter(adj.P.Val<0.05,abs(ratioFC)>1.05))
aagmex = nrow(aagmex0 %>% filter(adj.P.Val<0.05,abs(ratioFC)>1.05))

gt_dc = nrow(gtex_decode_fc)
dc_aa = nrow(decode_aagmex_fc)
gt_aa = nrow(gtex_aagmex_fc)
gt_cd_aa = nrow(all_fc)

pdf("Triple_Venn_diagram.pdf", height = 6, width = 6)
venn.plot <- draw.triple.venn(area1 = gtex, area2 = decode, area3 = aagmex,
  n12 = gt_dc, n23 = dc_aa, n13 = gt_aa,
  n123 = gt_cd_aa, category = c("GTEx", "deCODE", "AAGMEEx"),
  euler.d=F, scaled=F, overrideTriple=T)

dev.off()

save.image("deg.RData")
```

The result is illustrated in Figure 4.

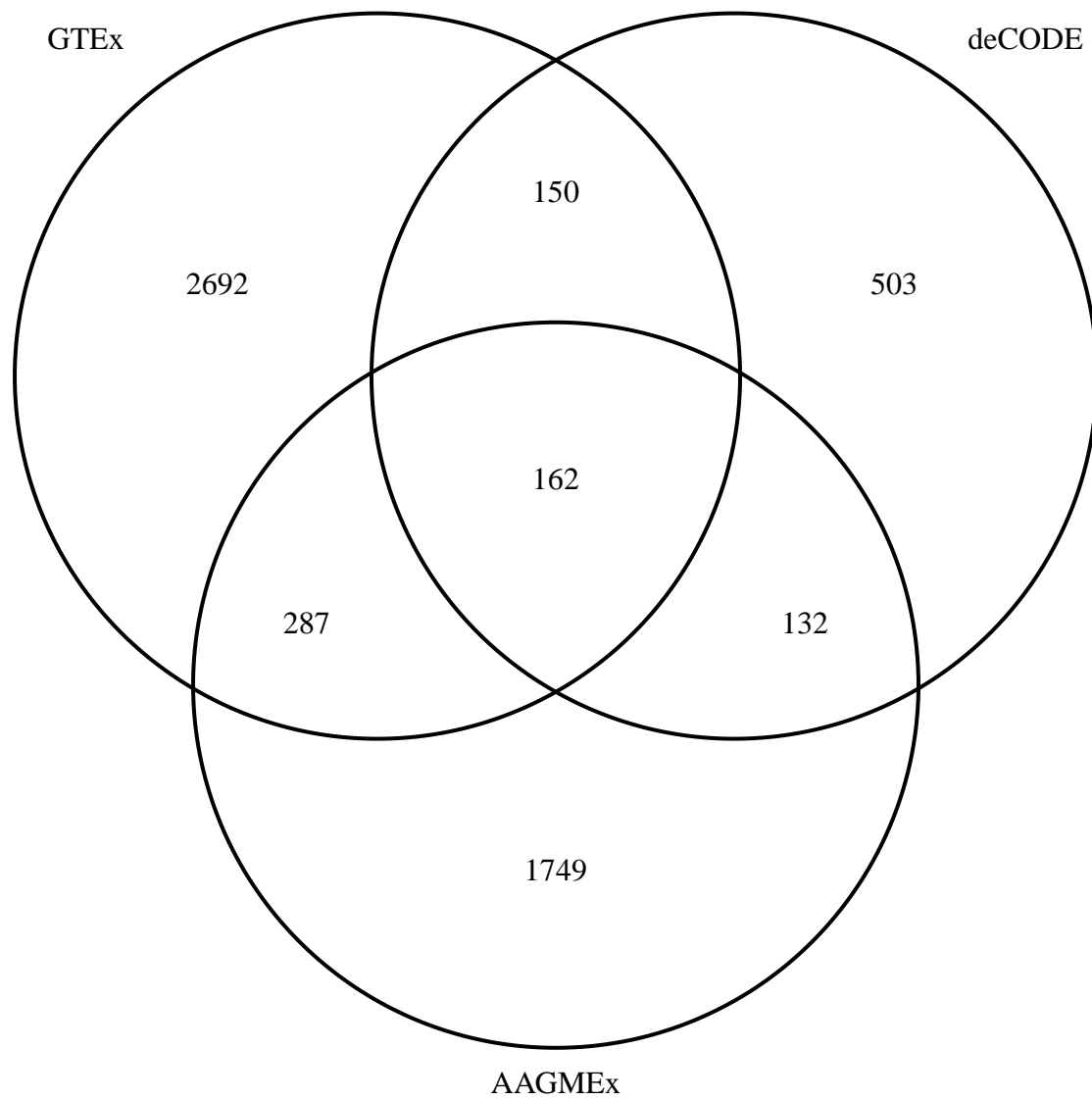


Figure 4: Venn-diagram delineating the differentially expressed genes in the three different data groups.