Videlock et al IBS microarray Code

(organized roughly by paper section heading)

packages needed
for information about installing WGCNA, see: https://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/Rpackages/WGCNA/

```
source("https://bioconductor.org/biocLite.R")
biocLite(c("annotate","limma","org.Hs.eg.db","Biobase",
           "hgu133plus2.db","ArrayExpress","affy", "impute", "GO.db", "preprocessCore",
           "AnnotationDbi","GOstats","hgu95av2.db","vsn"))

install.packages(c("gplots","ggplot2","WGCNA","NanoStringNorm"))
```

Microarray Differential Expression

Data set-up

Expression data

Using expression data already pre-processed by Agilent feature extraction. I used the file "mRNA Expression Profiling" which lists all targets where at least 10 out of 30 samples have flags in Present or Marginal. This data is "Processed 3" on Array Express.

Download:
https://www.ebi.ac.uk/arrayexpress/files/E-MTAB-5811/E-MTAB-5811.processed.3.zip

```
dat01<-read.delim("Normalized_unfiltered_mRNAonly.txt",row.names = 1)
```

Sample data

Download:
https://www.ebi.ac.uk/arrayexpress/files/E-MTAB-5811/E-MTAB-5811.sdrf.txt

```
pheno<-read.delim("E-MTAB-5811.sdrf.txt",row.names = 1)
```

reorder/reformat

```
pheno<-pheno[names(dat01),]
pheno$Sex<-ifelse(pheno$Characteristics.sex.=="male","M","F")
pheno$IBS<-ifelse(pheno$Characteristics.disease.=="irritable bowel syndrome","IBS","HC")
pheno$Group[pheno$Characteristics.clinical.history.=="healthy control"]<-"HC"
pheno$Group[pheno$Characteristics.clinical.history.=="constipation"]<-"C"
pheno$Group[pheno$Characteristics.clinical.history.=="diarrhea"]<-"D"
pheno<-pheno[,c("IBS","Group","Sex")]
```

Array Data

Download:
https://www.ebi.ac.uk/arrayexpress/files/A-MTAB-619/A-MTAB-619.adf.txt remove lines above column names

```r
datArray<-read.delim("A-MTAB-619.adf.txt",row.names = 1)
names(datArray)<-c("Seq","EntrezID")
```

match expression data, add symbol and gene name

```r
datArray<-datArray[rownames(dat01),]

library(annotate)
library(org.Hs.eg.db)

datArray$Symbol<-NA;datArray$Name<-NA

datArray2<-datArray[!is.na(datArray$EntrezID),]
datArray2$Symbol <- getSYMBOL(as.character(datArray2$EntrezID),"org.Hs.eg.db")
datArray2$Name <- as.character(lookUp(as.character(datArray2$EntrezID),
                                      "org.Hs.eg.db", "GENENAME"))

datArray3<-rbind(datArray[!rownames(datArray)%in%rownames(datArray2),],datArray2)

# match expression data
datArray3<-datArray3[rownames(dat01),c("Symbol","Name","EntrezID")]
```

compile expression set

```r
library(Biobase)
phenoData<-new("AnnotatedDataFrame", data=pheno)
UCmRNAeset<- new("ExpressionSet", exprs=as.matrix(dat01), phenoData=phenoData)
fData(UCmRNAeset)<-datArray3
save(UCmRNAeset,file = "UCmRNAeset.rda")
rm(list = ls())
```

Differential expression

```r
library(limma)
```

load expression set

```r
load("UCmRNAeset.rda")
```

convenience function to edit topTable (limma output)

```r
# adds means, direction, FC, variable that is direction + Entrez ID
top.revision<-function(toptable,eset,Group1,Group2){
        dat1<-toptable
        dat1$Direction[dat1$logFC>0]<-"Up"
        dat1$Direction[dat1$logFC<0]<-"Down"
        dat1$FC <- 2^(dat1$logFC)
        dat1$DirID<-paste(dat1$Direction,dat1$EntrezID)
        eset1<-eset[rownames(dat1),eset$Group%in%Group1]
        eset2<-eset[rownames(dat1),eset$Group%in%Group2]
        dat1$mean1 <- rowMeans(exprs(eset1))
        dat1$mean2 <- rowMeans(exprs(eset2))
        return(dat1)
}
save(top.revision,file = "top.revision.rda")
```

Run limma

```r
f<-as.factor(UCmRNAeset$Group)
design <- model.matrix(~ 0 + f)
colnames(design)<-levels(f)
contrasts <- makeContrasts((D+C)/2-HC, D-HC, C-HC, levels=design)
fitUC<-lmFit(UCmRNAeset,design)
fitUC <- contrasts.fit(fitUC, contrasts)
fitUC <- eBayes(fitUC)
coefficients<-colnames(fitUC$coefficients)
save(fitUC,coefficients,file = "UCfit.rda")
```

IBS vs HC

```r
Group1<-c("D","C");Group2<-c("HC")
Cont<-coefficients[1]
UC.IBSvHCall <- topTable(fitUC, coef=Cont, adjust.method="BH", number=Inf)
UC.IBSvHCall<-top.revision(UC.IBSvHCall,UCmRNAeset,Group1,Group2)
#filter by raw mean > 200 in either group
filter<-UC.IBSvHCall$mean1>=log2(200)|UC.IBSvHCall$mean2>=log2(200)
UC.IBSvHC.200<-UC.IBSvHCall[filter,]
```

IBS-D vs HC

```r
Group1<-c("D");Group2<-c("HC")
Cont<-coefficients[2]
UC.DvHCall <- topTable(fitUC, coef=Cont, adjust.method="BH", number=Inf)
UC.DvHCall<-top.revision(UC.DvHCall,UCmRNAeset,Group1,Group2)
#filter by raw mean > 200 in either group
filter<-UC.DvHCall$mean1>=log2(200)|UC.DvHCall$mean2>=log2(200)
UC.DvHC.200<-UC.DvHCall[filter,]
```

IBS-C vs HC

```r
Group1<-c("C");Group2<-c("HC")
Cont<-coefficients[3]
UC.CvHCall <- topTable(fitUC, coef=Cont, adjust.method="BH", number=Inf)
UC.CvHCall<-top.revision(UC.CvHCall,UCmRNAeset,Group1,Group2)
#filter by raw mean > 200 in either group
filter<-UC.CvHCall$mean1>=log2(200)|UC.CvHCall$mean2>=log2(200)
UC.CvHC.200<-UC.CvHCall[filter,]
```

ReAnnotate was used to align probes to exome ranges Process output from reAnnotate

```r
options(stringsAsFactors = F)
reAnnoAgil<-read.delim("probesAgil_exome_readAnnotation.txt",stringsAsFactors = FALSE)
hits.agil<-strsplit(reAnnoAgil$HIT,"\\;")
names(hits.agil)<-reAnnoAgil$X.PROBE_ID
load("hitsAgil.rda")
for (Probe in names(hits.agil)) {
        hits.agil[[Probe]]$AgilProbe<-Probe
}

hit.agilTable<-do.call(rbind,hits.agil)
# write.csv(hit.agilTable,"hitAgilTable.csv")
# save(hits.agil,file = "hitsAgil.rda")
```

review of alignments revealed that most probes were specific to one or more transcript

```r
hit.conflicts<-read.csv("hitAgilTableConflicts.csv",row.names = 1)
NewAnnotPrboes<-hit.conflicts$AgilProbe

ConflictProbes<-as.character(
        na.omit(hit.conflicts$AgilProbe[hit.conflicts$NonSpecific==1])) #43

AgilMatched<-data.frame(row.names = hit.conflicts$AgilProbe,
                        RefSeq_ReAnno=hit.conflicts$RefSeq2,
                        Symb_ReAnno=hit.conflicts$Symb2)
for (Probe in names(hits.agil[!names(hits.agil)%in%rownames(AgilMatched)])) {
        df.probe<-hits.agil[[Probe]]
        AgilMatched[Probe,"RefSeq_ReAnno"]<-
                paste(unique(df.probe$RefSeq),collapse = "/")
        AgilMatched[Probe,"Symb_ReAnno"]<-
                paste(unique(df.probe$Symb),collapse = "/")
}
# save(AgilMatched,file = "AgilMatched.rda")
```

retain by highest variance among nonspecific probes

```r
load("UCmRNAeset.rda")
library(WGCNA)
options(stringsAsFactors = FALSE)

# subset expression data
datExpr<-exprs(UCmRNAeset)[rownames(exprs(UCmRNAeset))%in%ConflictProbes,]

# collapse rows by max variance----
ProbeKey<-read.csv("Nonspecific.csv")
datExpr2<-collapseRows(datExpr,rowGroup = ProbeKey$Symb,
                       rowID = ProbeKey$AgilProbe, method ="maxRowVariance")
ExcludeProbes<-names(datExpr2$selectedRow[datExpr2$selectedRow==F])
# load("UC_CvHCtoptables.rda")
UCfilt<-UC.CvHCall$adj.P.Val<=0.05 &
        (UC.CvHCall$mean1>=log2(200)|UC.CvHCall$mean2>=log2(200))
UC<-UC.CvHCall[UCfilt,] #1270
UC.maxVar<-UC[!rownames(UC)%in%ExcludeProbes,] #1248


for (Probe in rownames(UC.maxVar)) {
        if(Probe%in%rownames(AgilMatched)){
                UC.maxVar[Probe,"RefSeq_ReAnno"]<-AgilMatched[Probe,"RefSeq_ReAnno"]
                UC.maxVar[Probe,"Symb_ReAnno"]<-AgilMatched[Probe,"Symb_ReAnno"]
        }
}

# 2 of top 100 only mapped to non-coding sequences
UC.filteredVar_noNC<-UC.filteredVar[
        !UC.filteredVar$RefSeq_ReAnno%in%c("NR_046338","NR_147125"),]
#save----
# save(UC.filteredVar,UC.filteredVar_noNC,file = "UC_CvHC_MaxVarFilter.rda")
# write.csv(UC.filteredVar_noNC,"UCmaxVarfiltered_noNC.csv")
```

heatmap

```r
library(gplots)
load("Top100Desc.rda")

esetTop100<-UCmRNAeset[rownames(Top100Desc),]
dim(esetTop100) #100 x 30

# re-order subjects by clustering within group
ReorderSubjects<-function(eset){
        hc<-eset$Group=="HC"; d<-eset$Group=="D";c<-eset$Group=="C"
        eset.hc<-eset[,hc]; eset.d<-eset[,d]; eset.c<-eset[,c]
        getInd<-function(eset.group){
                data1<-t(exprs(eset.group))
                dend<-as.dendrogram(hclust(dist(data1)))
                dend <- reorder(dend, rowMeans(data1))
                Ind<-rev(order.dendrogram(dend))
                return(Ind)
        }
        subs.c<-rownames(pData(eset.c[,getInd(eset.c)]))
        subs.d<-rownames(pData(eset.d[,getInd(eset.d)]))
        subs.hc<-rownames(pData(eset.hc[,getInd(eset.hc)]))
        subs<-c(subs.c,subs.d,subs.hc)
        eset2<-eset[,subs]
}

UC.grouped<-ReorderSubjects(esetTop100)
colors<-pData(UC.grouped)
colors$color[colors$Group=="HC"]<-"black"
colors$color[colors$Group=="C"]<-"red"
colors$color[colors$Group=="D"]<-"green"
patientcolorsUC_grouped<-as.character(colors$color)

exprs.dat<-exprs(UC.grouped)

# save(UC.grouped,exprs.dat,patientcolorsUC_grouped,file = "UCgrouped.rda")

pdf("Top100.pdf", width = 3.5, height = 7.7)
heatmap.2(exprs.dat, scale = "row", Colv = NA, dendrogram = "none",
          ColSideColors=patientcolorsUC_grouped,
          trace="none",
          col = topo.colors(50),
          lmat=rbind(c(0,4),c(0,1),c(3,2),c(0,5)),
          lwid = c(0.01,.5), lhei = c(0.01,.03, 2,.08),
          labRow = paste(Top100Desc$Symbol,Top100Desc$Description,sep = ", ") ,
          labCol = FALSE,
          cexRow =  .7 ,
          margins = c(0,17), key.title = NA, density.info = "none",
          offsetRow = 0,
          keysize = 0.25,
          key.par = list(mar=c(3.2,0,0.5,37.5),cex=0.3),
          key.xtickfun = function() {
                  breaks = pretty(parent.frame()$breaks)
```

```
                    breaks = seq(breaks[1],breaks[length(breaks)],2)
                    list(at = parent.frame()$scale01(breaks),
                         labels = breaks)
          }
)

dev.off()
```

Sensitivity analysis for sex

```
xy<-fData(UCmRNAeset)
xy<-xy[!is.na(xy$EntrezID),]
xy$chr<-as.character(lookUp(as.character(xy$EntrezID), "org.Hs.eg.db", "CHR"))
xy<-xy[xy$chr%in%c("X","Y"),]
noxy<-setdiff(rownames(exprs(UCmRNAeset)),rownames(xy))
eset21<-UCmRNAeset[noxy,]
```

repeat fit exlcuding x and y

```
f<-as.factor(eset21$Group)
design <- model.matrix(~ 0 + f)
colnames(design)<-levels(f)
contrasts <- makeContrasts((D+C)/2-HC, D-HC, C-HC, levels=design)
fitUC2<-lmFit(eset21,design)
fitUC2 <- contrasts.fit(fitUC2, contrasts)
fitUC2 <- eBayes(fitUC2)
coefficients<-colnames(fitUC2$coefficients)
Group1<-c("C");Group2<-c("HC")
Cont<-coefficients[3]
UC.CvHCall2 <- topTable(fitUC2, coef=Cont, adjust.method="BH", number=Inf)
UC.CvHCall2<-top.revision(UC.CvHCall2,eset21,Group1,Group2)
#filter by raw mean > 200 in either group
filter<-UC.CvHCall2$mean1>=log2(200)|UC.CvHCall2$mean2>=log2(200)
UC.CvHC.200.2<-UC.CvHCall2[filter,]
UC.CvHC.filtered2<-UC.CvHC.200.2[UC.CvHC.200.2$adj.P.Val<=0.05,]
#1146
```

Collapse by max variance

```
# subset expression data
datExpr<-exprs(UCmRNAeset)[rownames(exprs(UCmRNAeset))%in%ConflictProbes,]
# collapse rows by max variance----
ProbeKey<-read.csv("Nonspecific.csv")
datExpr2<-collapseRows(datExpr,rowGroup = ProbeKey$Symb
                        , rowID = ProbeKey$AgilProbe, method ="maxRowVariance")
ExcludeProbes<-names(datExpr2$selectedRow[datExpr2$selectedRow==F])
UC.maxVar2<-UC.CvHC.filtered2[!rownames(UC.CvHC.filtered2)%in%ExcludeProbes,] #1204
```

make expression set just of these 1204 DETs

```
esetsig<-eset21[rownames(UC.maxVar2),]
```

control for sex within these DEGs

```
BH<-factor(esetsig$Group,levels = c("HC","C","D"))
Sex<-factor(esetsig$Sex,levels = c("M","F"))
design <- model.matrix(~Sex*BH) # Sex+BH+Sex*BH
colnames(design)[2:4]<-c("SexF","C","D")
fit_BH.sex<-lmFit(esetsig,design)
fit_BH.sex <- eBayes(fit_BH.sex)
```

5 with significant sex effect

```
SigSex <- topTable(fit_BH.sex, coef="SexF", adjust.method="BH",p.value = 0.05)
# 5
```

2 with signifcant interaction effect (these overlap with the 5 with significant sex effect)

```
SigInt <- topTable(fit_BH.sex, coef="SexF:BHC", adjust.method="BH",p.value = 0.05)
# 2
```

For two with significant Sex and interaction efffects C Vs HC effect remains significant controlling for Sex interaction

```
SigC<-topTable(fit_BH.sex, coef="C", adjust.method="BH",number = Inf)
SigC[rownames(SigInt),]
```

Exclude interaction effect for remainder (was not sigficicant in other 3)

```
design <- model.matrix(~Sex+BH)
colnames(design)[2:4]<-c("SexF","C","D")
fit_BH.sex<-lmFit(esetsig,design)
fit_BH.sex <- eBayes(fit_BH.sex)
```

For all 5 with significant sex effect, effect of CvHC remains sigificant after controlling for sex

```
SigC2 <- topTable(fit_BH.sex, coef="C", adjust.method="BH",number = Inf)
SigC2[rownames(SigSex),]
```

save

```
save(UC.CvHCall,UC.CvHC.filtered,file = "UC_CvHCtoptables.rda")
rm(list=ls())
```

WGCNA

download:
https://www.ebi.ac.uk/arrayexpress/files/E-MTAB-5811/E-MTAB-5811.processed.2.zip

Set-up

```
dat01<-read.delim("Normalized_SignalFiltered_noXY.txt",row.names = 1)
mean.filter<-rowMeans(dat01)>=log2(200)
dat01<-dat01[mean.filter,]
datExpr<-as.data.frame(t(dat01))
```

Create modules

for choosing soft thresholding power, see: https://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/Rpackages/WGCNA

```r
library(WGCNA)
options(stringsAsFactors = FALSE)
allowWGCNAThreads() #if computer allows multi-threading
adjacency = adjacency(datExpr, power = 14)
TOM = TOMsimilarity(adjacency) #takes very long (30-60 min depending on RAM)
dissTOM = 1-TOM
geneTree = hclust(as.dist(dissTOM), method = "average");
rm(adjacency)
```

Plot the resulting clustering tree (dendrogram)

```r
plot(geneTree, xlab="", sub="",
    main = "Gene clustering on TOM-based dissimilarity", labels = FALSE,
    hang = 0.04)
```

Module identification using dynamic tree cut:

```r
minModuleSize = 30;
dynamicMods = cutreeDynamic(dendro = geneTree,
    distM = dissTOM, deepSplit = 2, pamRespectsDendro = FALSE,
    minClusterSize = minModuleSize)
```

Convert numeric labels into colors

```r
dynamicColors = labels2colors(dynamicMods)
```

Plot the dendrogram and colors underneath

```r
plotDendroAndColors(geneTree, dynamicColors,
    "Dynamic Tree Cut", dendroLabels = FALSE, hang = 0.03,
    addGuide = TRUE, guideHang = 0.05, main = "Gene dendrogram and
    module colors")
```

Merging of modules whose expression profiles are very similar

```r
MEList = moduleEigengenes(datExpr, colors = dynamicColors)
MEs = MEList$eigengenes
```

Calculate dissimilarity of module eigengenes

```r
MEDiss = 1-cor(MEs);
```

Cluster module eigengenes

```r
METree = hclust(as.dist(MEDiss), method = "average");
```

Plot the result

```r
plot(METree, main = "Clustering of module eigengenes", xlab = "", sub = "")
```

height cut of 0.25, corresponding to correlation of 0.75:

```r
MEDissThres = 0.25
plot(METree, main = "Clustering of module eigengenes",
     xlab = "", sub = "") +
        abline(h=MEDissThres, col = "red")
```

Call an automatic merging function

```r
merge = mergeCloseModules(datExpr, dynamicColors, cutHeight = MEDissThres, verbose = 3)
mergedColors = merge$colors;
mergedMEs = merge$newMEs

plotDendroAndColors(geneTree, cbind(dynamicColors, mergedColors),
    c("Dynamic Tree Cut", "Merged dynamic"), dendroLabels = FALSE,
    hang = 0.03, addGuide = TRUE, guideHang = 0.05)
```

Rename to moduleColors

```r
moduleColors = mergedColors
```

Construct numerical labels corresponding to the colors

```r
colorOrder = c("grey", standardColors(50));
moduleLabels = match(moduleColors, colorOrder)-1;
MEs = mergedMEs
```

Relating modules to external clinical traits

```r
load("UCmRNAeset.rda")
datTraits<-pData(UCmRNAeset)
datTraits$IBS<-ifelse(datTraits$IBS=="IBS",1,0)
datTraits$Sex<-ifelse(datTraits$Sex=="F",1,0)
datTraits$C[datTraits$Group=="HC"]<-0
datTraits$C[datTraits$Group=="C"]<-1
datTraits$D[datTraits$Group=="HC"]<-0
datTraits$D[datTraits$Group=="D"]<-1
datTraits<-datTraits[,c("IBS","C","D","Sex")]
nGenes = ncol(datExpr)
nSamples = nrow(datExpr)
MEs0 = moduleEigengenes(datExpr, moduleColors)$eigengenes
MEs = orderMEs(MEs0)
moduleTraitCor = cor(MEs, datTraits, use = "p")
moduleTraitPvalue = corPvalueStudent(moduleTraitCor, nSamples)
textMatrix =  paste(signif(moduleTraitCor, 2),
    "\n(", signif(moduleTraitPvalue, 1), ")", sep = "")
dim(textMatrix) = dim(moduleTraitCor)

pdf(file = "wgcnaplot.pdf", height = 16)
par(mar = c(6, 8.5, 3, 3));
labeledHeatmap(Matrix = moduleTraitCor, textMatrix = textMatrix,
    xLabels = names(datTraits), yLabels = names(MEs),
    ySymbols = names(MEs), colorLabels = FALSE,
    colors = blueWhiteRed(50), setStdMargins = FALSE,
    cex.text = 0.5, zlim = c(-1,1),
    main = paste("Module-trait relationships"))
dev.off()
```

Comparison with external data

Data Acquisition: Mayo

ten samples are in triplicate with one done 3 months later. Per reference, these were well correlated (no change in expression over 3 months). I will use all available, because there is no annotation as to which were the ones done later

```
library(ArrayExpress)
library(affy)
library(hgu133plus2.db)
Mayo <- ArrayExpress("E-TABM-176")
```

sample data

```
phMayo <- pData(Mayo)

# remove extra columns
phMayo <- phMayo[,c(3,5:6)]

# rename (IBS is IBS/HC, Group is C/D/HC)
colnames(phMayo)<-c("ID","IBS", "Group")

# rename
phMayo$IBS[phMayo$IBS=="irritable bowel syndrome"] <- "IBS"
phMayo$IBS[phMayo$IBS=="normal"] <- "HC"
phMayo$Group[phMayo$Group=="diarrhea-predominant"] <- "D"
phMayo$Group[phMayo$Group=="constipation-predominant"] <- "C"
phMayo$Group[phMayo$Group=="not applicable"] <- "HC"
Mayo@phenoData@data <- phMayo  # add back new pheno table

# shorten row and column names
rownames(pData(Mayo))<- gsub(".Hybridization","",
    gsub("jaerssen.prdbe.jnj.com.Tab2MAGE.","",
    rownames(pData(Mayo))))

colnames(exprs(Mayo))<- gsub(".Hybridization","",
    gsub("jaerssen.prdbe.jnj.com.Tab2MAGE.","",
    colnames(exprs(Mayo))))
```

Data are not preprocessed

```
dim(Mayo)
# Features  Samples
#  1354896      131
```

Pre-process, normalize, log transform with RMA

```
MayoNorm <- rma(Mayo)
dim(MayoNorm)
# Features  Samples
#    54675      131
```

```
genes<-data.frame(row.names = rownames(exprs(MayoNorm)))
library(annotate)
genes$Symbol <- getSYMBOL(as.character(rownames(genes)),"hgu133plus2.db")
```

```r
genes$Name <- as.character(lookUp(as.character(rownames(genes)),
                                  "hgu133plus2.db", "GENENAME"))
genes$EntrezID <- as.character(lookUp(as.character(rownames(genes)),
    "hgu133plus2.db", "ENTREZID"))
fData(MayoNorm)<-genes
```

Data Acquisition: Nottingham

```r
Nott <- ArrayExpress("E-GEOD-36701")
phNott <- Nott@phenoData@data
phNott <- phNott[,33:36]
colnames(phNott) <- c("Batch", "Group", "Sex", "ID")
phNott$Group[phNott$Group=="HV"]<-"HC"
phNott$Group[phNott$Group=="IBS-D"]<-"D"
phNott$Group[phNott$Group=="IBS-C"]<-"C"
phNott$Group[phNott$Group=="PIBD"]<-"PIBSCamp"

# The data are not pre-processed
dim(Nott)
#Features  Samples
#1354896      221
```

Pre-process, normalize, log transform with RMA

```r
NottNorm <- rma(Nott)
dim(NottNorm)
#Features  Samples
#54675      221
```

The experiment is described as two batches. Compare mean expression from the batches to see if experimental groups across batches can be compared. (Code not included but I also did without combining batches and not much different)

```r
dataAll<-exprs(NottNorm)
rowmeans <- rowMeans(t(dataAll))
ph1 <- cbind(phNott,rowmeans)
library(ggplot2)
ggplot(ph1,aes(Batch,rowmeans))+geom_bar(stat = "summary", fun.y = "mean")
# Means are similar
```

make expression set

```r
# there is a punctuation mark difference in the sample names between expression data
# and pheno data (they are the same but this makes them identical)
colnames(dataAll)<-rownames(phNott)
genes<-data.frame(row.names = rownames(dataAll))
genes$Symbol <- getSYMBOL(as.character(rownames(genes)),"hgu133plus2.db")
genes$Name <- as.character(lookUp(as.character(rownames(genes)),
                                  "hgu133plus2.db", "GENENAME"))
genes$EntrezID <- as.character(lookUp(as.character(rownames(genes)),
                                      "hgu133plus2.db", "ENTREZID"))
pData(NottNorm)<-phNott
fData(NottNorm)<-genes
```

save

```
save(MayoNorm,file = "MayoNorm.rda")
save(NottNorm,file="NottNorm.rda")
rm(list=ls())
```

Differential expression: Mayo

this uses the duplicateCorrelation function to average replicates. I also used means with similar results. note that duplicateCorrelation can take ~30 min

```
load("MayoNorm.rda")
load("top.revision.rda")

f <- as.factor(MayoNorm$Group)
design <- model.matrix(~ 0 + f)
colnames(design)=levels(f)
subID<-MayoNorm$ID
cor <- duplicateCorrelation(MayoNorm, design, block = subID)
fitMayo <- lmFit(MayoNorm, design, block = subID, correlation = cor$consensus.correlation)
contrasts <- makeContrasts((D+C)/2-HC, D-HC, C-HC, levels=design)
fitMayo <- contrasts.fit(fitMayo, contrasts)
fitMayo <- eBayes(fitMayo)
coefficients<-colnames(fitMayo$coefficients)

Group1<-c("C");Group2<-c("HC");Cont<-coefficients[3]
MayoCvHCdupCorAll<- topTable(fitMayo, coef=Cont, adjust.method="BH", number=Inf)
MayoCvHCdupCorAll<-top.revision(MayoCvHCdupCorAll,MayoNorm,Group1,Group2)
```

Differential Expression: Nottingham

this averages healthy controls over two batches. also did using only batch 1 which compared IBS-C, IBS-D and HC and similar results

```
load("NottNorm.rda")
f <- as.factor(NottNorm$Group)
design <- model.matrix(~ 0 + f)
colnames(design)=levels(f)
subID<-NottNorm$ID
cor <- duplicateCorrelation(NottNorm, design, block = subID)
fitNott <- lmFit(NottNorm, design, block = subID, correlation = cor$consensus.correlation)
contrasts <- makeContrasts((D+C+PIBS+PIBSCamp)/4-HC, (D+C)/2-HC,D-HC,C-HC,levels=design)
fitNott <- contrasts.fit(fitNott, contrasts)
fitNott <- eBayes(fitNott)
coefficients<-colnames(fitNott$coefficients)

Group1<-"C";Group2<-"HC";Cont<-coefficients[4]
NottCvHCdupCorAll <- topTable(fitNott, coef=Cont, adjust.method="BH", number=Inf)
NottCvHCdupCorAll<-top.revision(NottCvHCdupCorAll,NottNorm,Group1,Group2)
```

save

```
save(fitMayo,fitNott,file = "MayoNottfitdupcor.rda")
```

```r
save(MayoCvHCdupCorAll,NottCvHCdupCorAll,file="MayoNottCvHC.rda")
rm(list=ls())
```

Find common genes

ReAnnotate was used to align UC probes to exome ranges. Affymetrix annotation (2016) downloaded from Thermofisher

```r
options(stringsAsFactors = F)

affy<-read.csv("HG-U133_Plus_2.na36.annot copy.csv")
affy<-affy[!affy$RefSeq.Transcript.ID=="---",]
hits.affy0<-strsplit(affy$RefSeq.Transcript.ID," /// ")
names(hits.affy0)<-affy$Probe.Set.ID
hits.affy<-list()
for (Probe in names(hits.affy0)) {
        x<-hits.affy0[[Probe]]
        Remove<-c(x[grep("XM",x)],x[grep("XR",x)],x[grep("NR",x)])
        x<-x[!x%in%Remove]
        if(length(x)>0){hits.affy[[Probe]]<-x}
}

affyUniqueTx<-data.frame(AffyProbe=character(0),RefSeq=character(0))
affyMultiTx<-list()

for (Probe in names(hits.affy)) {
        x<-hits.affy[[Probe]]
        if(length(x)==1){
                affyUniqueTx<-rbind(affyUniqueTx,c(Probe,x))
        }else{affyMultiTx[[Probe]]<-x}
}
names(affyUniqueTx)<-c("AffyProbe","RefSeq")
# save(hits.affy,affyUniqueTx,affyMultiTx,file = "AffyAnnotData.rda")
```

Finds affy probes matching by common RefSeqIDs

```r
load("hitsagil.rda")
agil.RefSeqs0<-lapply(hits.agil, function(x){x[,"RefSeq"]})
agil.RefSeqs<-list()
for (Probe in names(agil.RefSeqs0)) {
        x<-agil.RefSeqs0[[Probe]]
        Remove<-x[grep("NR",x)]
        x<-x[!x%in%Remove]
        if(length(x)>0){agil.RefSeqs[[Probe]]<-x}
}

AgilUniqueTx<-data.frame(AgilProbe=character(0),RefSeq=character(0))
AgilMultiTx<-list()

for (Probe in names(agil.RefSeqs)) {
        x<-agil.RefSeqs[[Probe]]
        if(length(x)==1){
                AgilUniqueTx<-rbind(AgilUniqueTx,c(Probe,x))
        }else{AgilMultiTx[[Probe]]<-x}
}
names(AgilUniqueTx)<-c("AgilProbe","RefSeq")
# save(hits.agil,agil.RefSeqs,AgilMultiTx,AgilUniqueTx,file = "AgilAnnotData.rda")

CommonUniqueTx<-intersect(affyUniqueTx$RefSeq,AgilUniqueTx) #0

agil.matches<-lapply(agil.RefSeqs, function(x){
        y<-list()
        for (affyProbe in names(hits.affy)) {
                if(length(intersect(x,hits.affy[[affyProbe]]))>0){
                        y[[affyProbe]]<-intersect(x,hits.affy[[affyProbe]])
                }
        }
        return(y)
})

# save(agil.matches,file = "AgilMatches.rda")
```

Subset

```r
noMatch<-list(); SingleMatch<-list(); MultiMatch<-list()
for (Probe in names(agil.matches)) {
        if(length(agil.matches[[Probe]])==0){
                noMatch[[Probe]]<-agil.matches[[Probe]]
        }else if(length(agil.matches[[Probe]])==1){
                SingleMatch[[Probe]]<-agil.matches[[Probe]]
        }else {MultiMatch[[Probe]]<-agil.matches[[Probe]]}
}
length(noMatch); length(SingleMatch); length(MultiMatch)
# save(noMatch,SingleMatch,MultiMatch,file = "MatchCategoriesTx.rda")
```

NoMatch 331 SingleMatch 452 MultiMatch 472

Start Matched table

```r
AffyProbes<-unlist(lapply(SingleMatch, function(x){names(x)}))
RefSeqs<-unlist(lapply(SingleMatch, function(x){paste(unlist(x),collapse = "/")}))
Matched<-data.frame(AgilProbe=names(SingleMatch),
                    AffyProbe=AffyProbes,
                    Match=RefSeqs)
# save(Matched,file = "MatchedTx.rda")
```

Remove matched from multimatch – no difference

```r
MultiMatch2<-lapply(MultiMatch, function(x){
        x<-x[!names(x)%in%AffyProbes]})
```

Tabulate multimatches

```r
MultiMatch2<-lapply(MultiMatch2, function(x){
        x<-lapply(x, function(y){y<-paste(y,collapse = "/")})
})

for (Probe in names(MultiMatch2)) {
        df<-data.frame(AgilProbe=Probe,
                       AffyProbe=names(MultiMatch2[[Probe]]),
                       Match=unlist(MultiMatch2[[Probe]]))
        MultiMatch2[[Probe]]<-df
}

MultiMatchTable<-do.call(rbind,MultiMatch2)
# save(MultiMatch2,file = "MultiMatchTx2.rda")
# save(MultiMatchTable,file = "MultiMatchTxTable.rda")
# write.csv(MultiMatchTable,"MultiMatchTxTable.csv")
```

Unmatched by RefSeq - see if match by gene

```
UsedAffy<-c(MultiMatchTable$AffyProbe,Matched$AffyProbe)
UnusedAffyHits<-hits.affy[!names(hits.affy)%in%UsedAffy]
NoMatchAgil<-names(noMatch)
NoMatchAgilHits<-hits.agil[names(hits.agil)%in%NoMatchAgil]

agil.Genes<-lapply(NoMatchAgilHits, function(x){x[,"Symb"]})

genehits.affy0<-strsplit(affy$Gene.Symbol," /// ")
names(genehits.affy0)<-affy$Probe.Set.ID
# save(genehits.affy0,file = "genehitsAffy.rda")
genehits.affy<-genehits.affy0[names(genehits.affy0)%in%names(UnusedAffyHits)]

agil.matches.gene<-lapply(agil.Genes, function(x){
        y<-list()
        for (affyProbe in names(genehits.affy)) {
                if(length(intersect(x,genehits.affy[[affyProbe]]))>0){
                        y[[affyProbe]]<-intersect(x,genehits.affy[[affyProbe]])
                }
        }
        return(y)
})
# save(agil.matches.gene,file = "AgilMatchesGene.rda")

noMatchGene<-list()
SingleMatchGene<-list()
MultiMatchGene<-list()

for (Probe in names(agil.matches.gene)) {
        if(length(agil.matches.gene[[Probe]])==0){
                noMatchGene[[Probe]]<-agil.matches.gene[[Probe]]
        }else if(length(agil.matches.gene[[Probe]])==1){
                SingleMatchGene[[Probe]]<-agil.matches.gene[[Probe]]
        }else {MultiMatchGene[[Probe]]<-agil.matches.gene[[Probe]]}
}
length(noMatchGene);length(SingleMatchGene);length(MultiMatchGene)
# save(noMatchGene,SingleMatchGene,MultiMatchGene,file = "MatchCategoriesGene.rda")

AffyProbesGeneMatch<-unlist(lapply(SingleMatchGene, function(x){names(x)}))
Genes<-unlist(lapply(SingleMatchGene, function(x){paste(unlist(x),collapse = "/")}))
MatchedGene<-data.frame(AgilProbe=names(SingleMatchGene),
                    AffyProbe=AffyProbesGeneMatch,
                    Match=Genes)
# save(MatchedGene,file = "MatchedGene.rda")
```

No match 330 Single 1 Multi 0

Combined Key

```
MatchKey<-rbind(Matched,MultiMatchTable,MatchedGene)
rownames(MatchKey)<-NULL
matches.unique<-unique(MatchKey[,c("AgilProbe","AffyProbe")])
MatchKey<-MatchKey[rownames(matches.unique),]
# save(MatchKey,file = "MatchKey.rda")

# remove the UC probes excluded by max variance filter
load("UC_CvHC_MaxVarFilter.rda")
MatchKey<-MatchKey[MatchKey$AgilProbe%in%rownames(UC.filteredVar_noNC),]

NonUniqueMatches<-MatchKey[MatchKey$AffyProbe%in%
                                   MatchKey[duplicated(MatchKey$AffyProbe),"AffyProbe"],]

#add full list of matched RefSeqs and reannotated UC probes with isoform specificity

for (i in 1:nrow(NonUniqueMatches)) {
        AgilProbe<-NonUniqueMatches$AgilProbe[i]
        AffyProbe<-NonUniqueMatches$AffyProbe[i]
        probeRS<-hits.affy[[AffyProbe]]
        probeSymb<-genehits.affy0[[AffyProbe]]
        NonUniqueMatches$Symb[i]<-paste(unique(probeSymb),collapse = "/")
        NonUniqueMatches$RefSeqAll[i]<-paste(unique(probeRS),collapse = "/")
        NonUniqueMatches[NonUniqueMatches$AgilProbe==AgilProbe,"AgilSymb"]<-
                UC.maxMean_noNC[AgilProbe,"Symb_ReAnno"]
        NonUniqueMatches[NonUniqueMatches$AgilProbe==AgilProbe,"AgilRS"]<-
                UC.maxMean_noNC[AgilProbe,"RefSeq_ReAnno"]
}

# write.csv(NonUniqueMatches,"NonUniqueMatches.csv")
length(unique(NonUniqueMatches$AffyProbe))
#These resulted from 90 non-specific affy probes matching to two agil probes.
# In order to choose the correct match, collapse agil probes by max mean in UC data.
ConfToCollapse<-read.csv("NonUniqueMatchesResolved.csv")
load("UCmRNAeset.rda")
library(WGCNA)
options(stringsAsFactors = FALSE)
# subset expression data
datExpr<-exprs(UCmRNAeset)[rownames(exprs(UCmRNAeset))%in%ConfToCollapse$AgilProbe,]
datExpr2<-collapseRows(datExpr,rowGroup = ConfToCollapse$Symb,
                       rowID = ConfToCollapse$AgilProbe, method = "maxRowVariance")
ExcludeProbes<-names(datExpr2$selectedRow[datExpr2$selectedRow==F])

UCtoMatch<-UC.filteredVar_noNC[!rownames(UC.filteredVar_noNC)%in%ExcludeProbes,]

MatchKey2<-MatchKey[!MatchKey$AgilProbe%in%ExcludeProbes,]

# save(UCtoMatch,MatchKey2,file = "MatchDataMaxVar.rda")
```

Collapse External data by highest variance per matched probe among probes with multiple matches

```r
load("MatchDataMaxVar.rda")
library(WGCNA)
options(stringsAsFactors = FALSE)
```

Mayo

```r
load("MayoNottCvHC.rda")
load("MayoNorm.rda")

# collapse rows by max variance----
datExpr<-exprs(MayoNorm)
MatchTab<-MayoCvHCdupCorAll
MatchName<-"M"
datExpr2<-datExpr[rownames(datExpr)%in%MatchKey2$AffyProbe,]
datExpr2<-collapseRows(datExpr,rowGroup = MatchKey2$AgilProbe,
                       rowID = MatchKey2$AffyProbe, method = "maxRowVariance")
datExpr3<-datExpr2$datETcollapsed
rownames(datExpr3)<-names(datExpr2$selectedRow[datExpr2$selectedRow==T])
matchResult<-MatchKey2[MatchKey2$AffyProbe%in%
                               names(datExpr2$selectedRow[datExpr2$selectedRow==T]),]

for (i in 1:nrow(matchResult)) {
        AgilProbe<-matchResult$AgilProbe[i]
        AffyProbe<-matchResult$AffyProbe[i]
        matchResult[i,"Symb"]<-UCtoMatch[AgilProbe,"Symb_ReAnno"]
        matchResult[i,"logFC_UC"]<-UCtoMatch[AgilProbe,"logFC"]
        matchResult[i,"FC_UC"]<-UCtoMatch[AgilProbe,"FC"]
        matchResult[i,"p_UC"]<-UCtoMatch[AgilProbe,"P.Value"]
        matchResult[i,"FDR_UC"]<-UCtoMatch[AgilProbe,"adj.P.Val"]
        matchResult[i,"AveExpr_UC"]<-UCtoMatch[AgilProbe,"AveExpr"]
        matchResult[i,"mean1_UC"]<-UCtoMatch[AgilProbe,"mean1"]
        matchResult[i,"mean2_UC"]<-UCtoMatch[AgilProbe,"mean2"]
        matchResult[i,"Dir_UC"]<-UCtoMatch[AgilProbe,"Direction"]
        matchResult[i,paste("logFC",MatchName,sep = "_")]<-MatchTab[AffyProbe,"logFC"]
        matchResult[i,paste("FC",MatchName,sep = "_")]<-MatchTab[AffyProbe,"FC"]
        matchResult[i,paste("p",MatchName,sep = "_")]<-MatchTab[AffyProbe,"P.Value"]
        matchResult[i,paste("FDR",MatchName,sep = "_")]<-MatchTab[AffyProbe,"adj.P.Val"]
        matchResult[i,paste("AveExpr",MatchName,sep = "_")]<-MatchTab[AffyProbe,"AveExpr"]
        matchResult[i,paste("mean1",MatchName,sep = "_")]<-MatchTab[AffyProbe,"mean1"]
        matchResult[i,paste("mean2",MatchName,sep = "_")]<-MatchTab[AffyProbe,"mean2"]
        matchResult[i,paste("Dir",MatchName,sep = "_")]<-MatchTab[AffyProbe,"Direction"]
        matchResult[i,"Name"]<-UCtoMatch[AgilProbe,"Name"]
}
rownames(matchResult)<-matchResult$AgilProbe
dir.filter<-matchResult[,"Dir_UC"]==matchResult[,paste("Dir",MatchName,sep = "_")]
matchedDir<-matchResult[dir.filter,]
matchedDirSig<-matchedDir[matchedDir[,paste("p",MatchName,sep = "_")]<0.05,]
#write.csv(matchedDirSig,paste(MatchName,"_MatchedSig.csv",sep = ""))
#write.csv(matchedDir,paste(MatchName,"_MatchedDir.csv",sep = ""))

M.datExpr.matched<-datExpr3
M.CvHC.matchedDir<-matchedDir
```

```
M.CvHC.matchedDirSig<-matchedDirSig
# save(M.datExpr.matched,M.CvHC.matchedDir,M.CvHC.matchedDirSig,file = "Mayo_Matched.rda")
```

Nottingham

```
load("MatchDataMaxVar.rda")
load("MayoNottCvHC.rda")
load("NottNorm.rda")
datExpr<-exprs(NottNorm)
MatchTab<-NottCvHCdupCorAll
MatchName<-"N"
datExpr2<-datExpr[rownames(datExpr)%in%MatchKey2$AffyProbe,]
datExpr2<-collapseRows(datExpr,rowGroup = MatchKey2$AgilProbe,
                       rowID = MatchKey2$AffyProbe, method = "maxRowVariance")
datExpr3<-datExpr2$datETcollapsed
rownames(datExpr3)<-names(datExpr2$selectedRow[datExpr2$selectedRow==T])
matchResult<-MatchKey2[MatchKey2$AffyProbe%in%
                           names(datExpr2$selectedRow[datExpr2$selectedRow==T]),]

for (i in 1:nrow(matchResult)) {
        AgilProbe<-matchResult$AgilProbe[i]
        AffyProbe<-matchResult$AffyProbe[i]
        matchResult[i,"Symb"]<-UCtoMatch[AgilProbe,"Symb_ReAnno"]
        matchResult[i,"logFC_UC"]<-UCtoMatch[AgilProbe,"logFC"]
        matchResult[i,"FC_UC"]<-UCtoMatch[AgilProbe,"FC"]
        matchResult[i,"p_UC"]<-UCtoMatch[AgilProbe,"P.Value"]
        matchResult[i,"FDR_UC"]<-UCtoMatch[AgilProbe,"adj.P.Val"]
        matchResult[i,"AveExpr_UC"]<-UCtoMatch[AgilProbe,"AveExpr"]
        matchResult[i,"mean1_UC"]<-UCtoMatch[AgilProbe,"mean1"]
        matchResult[i,"mean2_UC"]<-UCtoMatch[AgilProbe,"mean2"]
        matchResult[i,"Dir_UC"]<-UCtoMatch[AgilProbe,"Direction"]
        matchResult[i,paste("logFC",MatchName,sep = "_")]<-MatchTab[AffyProbe,"logFC"]
        matchResult[i,paste("FC",MatchName,sep = "_")]<-MatchTab[AffyProbe,"FC"]
        matchResult[i,paste("p",MatchName,sep = "_")]<-MatchTab[AffyProbe,"P.Value"]
        matchResult[i,paste("FDR",MatchName,sep = "_")]<-MatchTab[AffyProbe,"adj.P.Val"]
        matchResult[i,paste("AveExpr",MatchName,sep = "_")]<-MatchTab[AffyProbe,"AveExpr"]
        matchResult[i,paste("mean1",MatchName,sep = "_")]<-MatchTab[AffyProbe,"mean1"]
        matchResult[i,paste("mean2",MatchName,sep = "_")]<-MatchTab[AffyProbe,"mean2"]
        matchResult[i,paste("Dir",MatchName,sep = "_")]<-MatchTab[AffyProbe,"Direction"]
        matchResult[i,"Name"]<-UCtoMatch[AgilProbe,"Name"]
}
rownames(matchResult)<-matchResult$AgilProbe
dir.filter<-matchResult[,"Dir_UC"]==matchResult[,paste("Dir",MatchName,sep = "_")]
matchedDir<-matchResult[dir.filter,]
matchedDirSig<-matchedDir[matchedDir[,paste("p",MatchName,sep = "_")]<0.05,]
#write.csv(matchedDirSig,paste(MatchName,"_MatchedSig.csv",sep = ""))
#write.csv(matchedDir,paste(MatchName,"_MatchedDir.csv",sep = ""))

N.datExpr.matched<-datExpr3
N.CvHC.matchedDir<-matchedDir
N.CvHC.matchedDirSig<-matchedDirSig
# save(N.datExpr.matched,N.CvHC.matchedDir,N.CvHC.matchedDirSig,file = "Nott_Matched.rda")
```

All 3

```r
load("Nott_Matched.rda")
load("Mayo_Matched.rda")
commonSymb<-intersect(M.CvHC.matchedDirSig$Symb,N.CvHC.matchedDirSig$Symb)
commonTable<-M.CvHC.matchedDirSig[M.CvHC.matchedDirSig$Symb%in%commonSymb,]
nottCommon<-N.CvHC.matchedDirSig[rownames(commonTable),]
notCols<-names(nottCommon)[grep("_N",names(nottCommon))]
commonTable<-cbind(commonTable,nottCommon[,names(nottCommon)%in%notCols])
# write.csv(commonTable,"CommonGeneTable.csv")
# save(commonTable,file = "commonTable.rda")
```

heatmap

```r
load("UCmRNAeset.rda")
library(gplots)

load("CommonDesc.rda")

esetCommon<-UCmRNAeset[rownames(CommonDesc),]
dim(esetCommon) #57 x 30

# re-order subjects by clustering within group
ReorderSubjects<-function(eset){
        hc<-eset$Group=="HC"; d<-eset$Group=="D";c<-eset$Group=="C"
        eset.hc<-eset[,hc]; eset.d<-eset[,d]; eset.c<-eset[,c]
        getInd<-function(eset.group){
                data1<-t(exprs(eset.group))
                dend<-as.dendrogram(hclust(dist(data1)))
                dend <- reorder(dend, rowMeans(data1))
                Ind<-rev(order.dendrogram(dend))
                return(Ind)
        }
        subs.c<-rownames(pData(eset.c[,getInd(eset.c)]))
        subs.d<-rownames(pData(eset.d[,getInd(eset.d)]))
        subs.hc<-rownames(pData(eset.hc[,getInd(eset.hc)]))
        subs<-c(subs.c,subs.d,subs.hc)
        eset2<-eset[,subs]
}

common.grouped<-ReorderSubjects(esetCommon)
colors<-pData(common.grouped)
colors$color[colors$Group=="HC"]<-"black"
colors$color[colors$Group=="C"]<-"red"
colors$color[colors$Group=="D"]<-"green"
patientcolorsCommon_grouped<-as.character(colors$color)

exprs.dat<-exprs(common.grouped)

# save(common.grouped,exprs.dat,patientcolorsCommon_grouped,file = "Commongrouped.rda")

# pdf("CommonHeatmap.pdf", width = 3.5, height = 5)
heatmap.2(exprs.dat, scale = "row", Colv = NA, dendrogram = "none",
          ColSideColors=patientcolorsUC_grouped,
          trace="none",
          col = topo.colors(50),
          lmat=rbind(c(0,4),c(0,1),c(3,2),c(0,5)),
          lwid = c(0.01,.5), lhei = c(0.01,.03, 1.3,.08),
          labRow = paste(CommonDesc$Symb,CommonDesc$Description2,sep = ", ") ,
          labCol = FALSE,
          cexRow =  .7 ,
          margins = c(0,17), key.title = NA, density.info = "none",
          offsetRow = 0,
          keysize = 0.25,
          key.par = list(mar=c(3.2,0,0.5,37.5),cex=0.3),
          key.xtickfun = function() {
                  breaks = pretty(parent.frame()$breaks)
                  breaks = seq(breaks[1],breaks[length(breaks)],2)
                  list(at = parent.frame()$scale01(breaks),
                       labels = breaks)
          }
)

# dev.off()
```

```
rm(list=ls())
```

GO enrichment

```
library(GOstats)
library(hgu95av2.db)
view.GOstats<-function(toptable,universe,dir){
        if(dir=="up"){toptable<-toptable[toptable$logFC>0,]}
        else{toptable<-toptable[toptable$logFC<0,]}
        selectedEntrezIds<-as.character(toptable$EntrezID)
        params <- new("GOHyperGParams", geneIds = selectedEntrezIds,
        universeGeneIds = universe, annotation = "hgu95av2.db",
        ontology = "BP", pvalueCutoff = 0.05, conditional = FALSE,
        testDirection = "over")
        hgOver <- hyperGTest(params)
        table <- summary(hgOver)
        return(table)
}
```

common genes

```
load("UCfit.rda")
load("MayoNottfitdupcor.rda")
load("Mayo_Matched.rda")
load("MatchDataMaxVar.rda")
commonTable<-UCtoMatch[rownames(UCtoMatch)%in%rownames(M.CvHC.matchedDirSig),]
universeUC<-as.character(fitUC$genes$EntrezID)
universeMayo<-as.character(fitMayo$genes$EntrezID)
universeCommon<-union(universeUC,universeMayo) #same chip/same universe for Roch and Nott
common.up<-view.GOstats(commonTable,universeCommon,"up")
common.down<-view.GOstats(commonTable,universeCommon,"down")
# write.csv(common.up[common.up$Size>=10&common.up$Count>1,],"commmon_up.csv")
# write.csv(common.down[common.down$Size>=10&common.down$Count>1,],"commmon_down.csv")
```

WGCNA module done in same way, universe was EntrezIDs in WGCNA analysis.

```
rm(list=ls())
```

Nanostring nCounter validation

quality control initially identified 4 outliers. The fourth is only z score 3.03 so will retain this one

```
library(NanoStringNorm)
```

load data (outliers removed)

```
load("NSraw0717NoOutliers.rda")
load("NSpheno0717NoOutliers.rda")
```

Normalization

See NanoStringNorm paper: https://doi.org/10.1093/bioinformatics/bts188

Code-count normalization: accounts for lane-by-lane variation of the nCounter platform. Normalization occurs by summarizing (i.e. mean, median, sum or geometric mean) the positive control counts and adjusting samples by a factor relative to other samples.

Background correction: negative controls

Sample content normalization: normalization genes summarized (e.g. geometric mean, etc.) and a sample adjustment factor is calculated.

convenience function for normalization

```
myNSnorm<- function(raw,method,logged){
        library(NanoStringNorm)
        norm <- NanoStringNorm(
                x = raw,
                CodeCount = method[1],
                Background = method[2],
                SampleContent = method[3],
                OtherNorm = method[4],
                round.values = TRUE,
                take.log = logged,
                return.matrix.of.endogenous.probes = TRUE)
        return(norm)
}
```

convenience function to save counts

```
savecounts<-function(countmatrix,namestart){
        date <- gsub("-","",gsub("2016-","",Sys.Date()))
        methName <- gsub("geomean","gm", gsub("\\.","", gsub("none","na",
        gsub("housekeeping","hk", paste(method,collapse = "_")))))
        filename <- paste(namestart,"_",methName,date,".csv",sep = "")
        write.csv(countmatrix, file = filename)
        print(paste("Filename:",filename))
        method.sum<<-methName
}
```

negative controls were extremely low and the NanoString rep advised against using a background control factor. for content norm, used low.cv.geo.mean which chooses good reference genes (low variation) since pre-selected reference genes (GAPDH, beta actin, PPIA had higher CV)

since will use voom, do not log transform

```
method <- c("geo.mean","none", "low.cv.geo.mean", "none")
NSnormNolog<-myNSnorm(raw = NSraw0717NoOutliers, method = method,logged = FALSE)
savecounts(countmatrix = NSnormNolog, namestart = "NSnolog45")
```

```
#there were 6 in the set that were not part of this analysis
load("NSvalidationMAprobes.rda") #maps symbols to microarray probes
NSnormNolog<-NSnormNolog[as.character(NSvalidationMAprobes$GeneSymbol),]
dim(NSnormNolog) #[1] 67 45
```

Differential Expression

voom to transform for linear modeling

```
library(limma)
f <- as.factor(NSpheno0717NoOutliers$Group)
design <- model.matrix(~ 0 + f)
levels(f)
## [1] "C"  "D"  "HC"
colnames(design)=levels(f)
v1<-voom(NSnormNolog,design)
```

Linear modeling

```
fit <- lmFit(v1, design)
contrasts <- makeContrasts((D+C)/2-HC, D-HC, C-HC, levels=design)
fitcon <- contrasts.fit(fit, contrasts)
eb1 <- eBayes(fitcon)
coefficients<-colnames(eb1$coefficients)
```

convenience function to save results

```
mytop<-function(object,names){
        # returns list, saves csv of top tables
        # object: eg eb
        # names vector of names of top tables
        toplist<-list()
        top1<-topTable(object, coef = coefficients[1], adjust.method = "BH", number = Inf)
        top2<-topTable(object, coef = coefficients[2], adjust.method = "BH", number = Inf)
        top3<-topTable(object, coef = coefficients[3], adjust.method = "BH", number = Inf)
        write.csv(top1,file = paste(names[1],"csv",sep = "."))
        write.csv(top2,file = paste(names[2],"csv",sep = "."))
        write.csv(top3,file = paste(names[3],"csv",sep = "."))
        toplist[["IBSvHC"]] <-top1
        toplist[["DvHC"]] <-top2
        toplist[["CvHC"]] <-top3
        return(toplist)
}
```

```
tops_all<-mytop(eb1,names = c("IBSvHCall","DvHCall","CvHCall"))
```

determine results in microarray cohort

```
maCohort<-rownames(NSpheno0717NoOutliers[NSpheno0717NoOutliers$Cohort=="MA",])
NS.ma<-NSnormNolog[,colnames(NSnormNolog)%in%maCohort]
phenoNS.ma<-NSpheno0717NoOutliers[colnames(NS.ma),]
f <- as.factor(phenoNS.ma$Group)
design <- model.matrix(~ 0 + f)
levels(f)
## [1] "C"  "D"  "HC"
colnames(design)=levels(f)
v1<-voom(NS.ma,design)
fit <- lmFit(v1, design)
contrasts <- makeContrasts((D+C)/2-HC, D-HC, C-HC, levels=design)
fitcon <- contrasts.fit(fit, contrasts)
eb1 <- eBayes(fitcon)
coefficients<-colnames(eb1$coefficients)
NStopsMA<-mytop(eb1,names = c("maIBSvHCall","maDvHCall","maCvHCall"))
```

validation cohort alone

```
valCohort<-rownames(NSpheno0717NoOutliers[NSpheno0717NoOutliers$Cohort=="Validation",])
NS.val<-NSnormNolog[,colnames(NSnormNolog)%in%valCohort]
phenoNS.val<-NSpheno0717NoOutliers[colnames(NS.val),]
f <- as.factor(phenoNS.val$Group)
design <- model.matrix(~ 0 + f)
levels(f)
## [1] "C"  "D"  "HC"
colnames(design)=levels(f)
v1<-voom(NS.val,design)
fit <- lmFit(v1, design)
contrasts <- makeContrasts((D+C)/2-HC, D-HC, C-HC, levels=design)
fitcon <- contrasts.fit(fit, contrasts)
eb1 <- eBayes(fitcon)
coefficients<-colnames(eb1$coefficients)
NStopsVal<-mytop(eb1,names = c("valIBSvHCall","valDvHCall","valCvHCall"))
```

validation requires same direction of fold change

```
load("UC_CvHCtoptables.rda")
MA.C<-UC.CvHCall
MA.C<-MA.C[rownames(NSvalidationMAprobes),]
names(MA.C)<-paste(names(MA.C),"MA",sep = "_")
rownames(MA.C)<-MA.C$Symbol_MA

NS.C.all<-tops_all$CvHC
NS.C.MA<-NStopsMA$CvHC
NS.C.val<-NStopsVal$CvHC

names(NS.C.all)<-paste(names(NS.C.all),"NS",sep = "_")
names(NS.C.MA)<-paste(names(NS.C.MA),"NS",sep = "_")
names(NS.C.val)<-paste(names(NS.C.val),"NS",sep = "_")
```

```r
length(setdiff(rownames(MA.C),rownames(NS.C.all))) #0

MA.C<-MA.C[order(rownames(MA.C)),]
NS.C.all<-NS.C.all[order(rownames(NS.C.all)),]
NS.C.MA<-NS.C.MA[order(rownames(NS.C.MA)),]
NS.C.val<-NS.C.val[order(rownames(NS.C.val)),]

NSMA.C.all<-cbind.data.frame(NS.C.all,MA.C)
NSMA.C.MA<-cbind.data.frame(NS.C.MA,MA.C)
NSMA.C.val<-cbind.data.frame(NS.C.val,MA.C)

matchDirection<-function(data){
        data<-data[data$P.Value_NS<=0.05,]
        same<-(data$logFC_MA>0&data$logFC_NS>0)|(data$logFC_MA<0&data$logFC_NS<0)
        data<-data[same==TRUE,]
}

NS.C.validated.all<-matchDirection(NSMA.C.all)
NS.C.validated.ma<-matchDirection(NSMA.C.MA)
NS.C.validated.val<-matchDirection(NSMA.C.val)

NS.C.validated.all.FDR<-NS.C.validated.all[NS.C.validated.all$adj.P.Val_NS<0.051,]
NS.C.validated.ma.FDR<-NS.C.validated.ma[NS.C.validated.ma$adj.P.Val_NS<0.051,]
NS.C.validated.val.FDR<-NS.C.validated.val[NS.C.validated.val$adj.P.Val_NS<0.051,]

NSMAvalC<-list()
NSMAvalC$allP<-NS.C.validated.all
NSMAvalC$maP<-NS.C.validated.ma
NSMAvalC$valP<-NS.C.validated.val
NSMAvalC$allFDR<-NS.C.validated.all.FDR
NSMAvalC$maFDR<-NS.C.validated.ma.FDR

for(df in names(NSMAvalC)){
        f<-paste("NSMAvalC",df,".csv",sep = "")
        write.csv(NSMAvalC[[df]],f)
}
```

save

```r
save(NSMAvalC,file = "NSMAvalC.rda")
save(NSnormNolog,file = "NSnormNolog67x45.rda")
save(tops_all,NStopsMA,NStopsVal,file = "NanoStringTopTables.rda")
rm(list=ls())
```

Association with symptoms

```r
load("NStraits.rda") #loads pheno data
load("NSnormNolog67x45.rda")
NSvoom<-voom(NSnormNolog)
traits<-traits[traits$IBS=="IBS",]
NS.norm.values<-as.data.frame(t(NSvoom$E))
geneData<-NS.norm.values[rownames(traits),]
```

```r
genes<-names(geneData)
```

Overall symptoms

```r
OverallSx<-data.frame(
        row.names = genes, F.pvalue=numeric(length(genes)),
        B=numeric(length(genes)),  Sx.pvalue=numeric(length(genes)))

for (i in 1:nrow(OverallSx)) {
        df1<-cbind(geneData[,rownames(OverallSx)[i]],traits)
        names(df1)[1]<-"gene"
        fit<-lm(BSQ_OverallSx~factor(Sex)+gene,data = df1, na.action = na.exclude)
        sumfit<-summary(fit)
        OverallSx[i,"F.pvalue"]<- pf(sumfit$fstatistic[1], sumfit$fstatistic[2],
                                     sumfit$fstatistic[3],lower.tail = FALSE)
        OverallSx[i,"B"]<-sumfit$coefficients["gene","Estimate"]
        OverallSx[i,"Sx.pvalue"]<-sumfit$coefficients["gene","Pr(>|t|)"]
}

OverallSx$FDR<-p.adjust(OverallSx$Sx.pvalue,method = "BH")
OverallSx<-OverallSx[order(OverallSx$Sx.pvalue),]
write.csv(OverallSx,"OverallSxregressions073117.csv")
```

Abdominal Pain

```r
Pain<-data.frame(
        row.names = genes, F.pvalue=numeric(length(genes)),
        B=numeric(length(genes)), Sx.pvalue=numeric(length(genes)))

for (i in 1:nrow(Pain)) {
        df1<-cbind(geneData[,rownames(Pain)[i]],traits)
        names(df1)[1]<-"gene"
        fit<-lm(BSQ_AbdPain~Age+factor(Sex)+BMI+gene,data = df1, na.action = na.exclude)
        sumfit<-summary(fit)
        Pain[i,"F.pvalue"]<-pf(sumfit$fstatistic[1], sumfit$fstatistic[2],
                               sumfit$fstatistic[3],lower.tail = FALSE)
        Pain[i,"B"]<-sumfit$coefficients["gene","Estimate"]
        Pain[i,"Sx.pvalue"]<-sumfit$coefficients["gene","Pr(>|t|)"]
}

Pain$FDR<-p.adjust(Pain$Sx.pvalue,method = "BH")
Pain<-Pain[order(Pain$Sx.pvalue),]
write.csv(Pain,"Painregressions073117.csv")
```

Bloating

```r
Bloating<-data.frame(
        row.names = genes,F.pvalue=numeric(length(genes)),
        B=numeric(length(genes)),Sx.pvalue=numeric(length(genes)))

for (i in 1:nrow(Bloating)) {
        df1<-cbind(geneData[,rownames(Bloating)[i]],traits)
        names(df1)[1]<-"gene"
        fit<-lm(BSQ_Bloating~gene,data = df1, na.action = na.exclude)
        sumfit<-summary(fit)
        Bloating[i,"F.pvalue"]<-pf(sumfit$fstatistic[1], sumfit$fstatistic[2],
                                   sumfit$fstatistic[3],lower.tail = FALSE)
        Bloating[i,"B"]<-sumfit$coefficients["gene","Estimate"]
        Bloating[i,"Sx.pvalue"]<-sumfit$coefficients["gene","Pr(>|t|)"]
}

Bloating$FDR<-p.adjust(Bloating$Sx.pvalue,method = "BH")
Bloating<-Bloating[order(Bloating$Sx.pvalue),]
write.csv(Bloating,"Bloatingregressions073117.csv")
```