- **Individual level differential analysis**

This analysis was performed by my colleague Liang He, who provided the following code: code for the Poisson mixed model.

First without RUV, for each cell type

```
library(lme4)
re <- glmer(formula = y ~ ad +offset(log(lib)) + (1 | ind), family = poisson, data = dt, nAGQ=10)
```

where y is the expression count for cell i and gene j, ad is a 0/1 indicator of AD, lib is library size for the cell, ind is the individual ID, dt is a data frame containing the count matrix of all cells in one cell type, nAGQ is the number of nodes in Gauss quadrature.

I found the Poisson model was better fitted for most genes in our data based on AIC than zero inflated Poisson mixed and NB mixed regression fitted using the following code

```
re <- glmer.nb(formula = y ~ ad +offset(log(lib)) + (1 | ind), data = dt)
library(glmmTMB)
re <- glmmTMB(y~ad + offset(log(lib))+(1|ind),data=dt,ziformula=~1,family=poisson)
```

The model with RUV is the same as above except that RUV covariates were added in the regression.
The code for obtaining the RUV covariates,

```
library(DESeq2)
library(RUVSeq)
d_e <- DGEList(data_ind, genes=rownames(data_ind))
keep <- rowSums(cpm(d_e)>1) >= 3
d_e <- d_e[keep, , keep.lib.sizes=FALSE]
d_e <- calcNormFactors(d_e, method="TMM")
design <- model.matrix(~ad)
d_e <- estimateGLMCommonDisp(d_e, design)
d_e <- estimateGLMTagwiseDisp(d_e, design)
fit1 <- glmFit(d_e, design)
res1 <- residuals(fit1, type="deviance")
ruvn <- 10
ruv_cov <- RUVr(round(d_e$counts), as.character(rownames(d_e$counts)), k=ruvn, res1)
```

, where data_ind is an individual-level count matrix by aggregating all cells within each cell type and each individual.

If you have further questions on this, please feel free to address Liang (liang.he@duke.edu) directly.

- **cell level differential analysis**

You can try this older function in R.

```
##############################################################################
# old function
scRNA.diff.run.wilcox.test<- function(QInexp, QIngroup) {
  group <- QIngroup
  # remove genes that aren't expressed in at least 10 cells
  QInexp <- QInexp[apply(QInexp > 0, 1, sum) > 10,]
  # Library size normalization
  lib_size = apply(QInexp, 2, sum)
  QInexp <- t(t(QInexp)/lib_size * median(lib_size))

  pVals <- apply(
    QInexp, 1, function(x) {
      wilcox.test(
        x[group == unique(QIngroup)[1]],
        x[group == unique(QIngroup)[2]]
      )$p.value
    }
  )
  # multiple testing correction
  pVals <- p.adjust(pVals, method = "fdr")

  temp.mean <- do.call(cbind, lapply(unique(group), function(i) apply(QInexp[,which(group == i)],
1, mean)))
  colnames(temp.meadians) <- unique(group)
  temp.out <- data.frame(gene.name=rownames(QInexp), adj.pvals=pVals, temp.mean,
FC=log2(temp.mean[,1])-log2(temp.mean[,2]))
  temp.out <- temp.out[order(temp.out$adj.pvals, decreasing = F),]
  return(temp.out)
}
##############################################################################
QInexp -- count matrix with the data corresponding to a given cell type.
QIngroup --  should be a vector specifying the pathology, no pathology groups of the columns in
QInexp.
The normalization is done internally, after removing cells with no detected expression.
The direction of the fold change will depend on your definition of the QIngroup vector, so you
should check afterwards by looking at the computed mean values of each group in the output.
Be aware that this function is not very efficient for a large number of cells -- it takes some time to
run.
##############################################################################
```

In our current studies we now use this function that takes advantage of the package presto and is way faster::

```
##############################################################################

Wilcox.differential <- function(QInexp, QIngroup) {
  A = as(QInexp, 'dgTMatrix')
  cs = Matrix::colSums(A)
  norm.out = Matrix::sparseMatrix(i = A@i+1, j = A@j+1, x = log(1 + median(cs)*(A@x / cs[A@j
+ 1])), dims = dim(A))
  rownames(norm.out) <- rownames(A)

  o <- presto::wilcoxauc(norm.out, QIngroup)
  o <- o[o$group==1,]
  o <- o[order(o$auc, decreasing = T),]
  rownames(o) <- o$feature
  o$class.power <- (abs(o$auc-0.5)) * 2
  return(o)
}
##############################################################################
QInexp -- count matrix with the data corresponding to a given cell type.
QIngroup --  should be a BINARY vector specifying the pathology, no pathology groups of the
columns in QInexp -- for instance if pathology 1 otherwise 0.
Normalization is performed internally
##############################################################################
```