Data filtering for Heaney et al. 2017

Sanjeev V Namjoshi (snamjoshi87@utexas.edu)

February 23rd, 2017 (revised June 15th, 2019)

This document contains all the code needed to generate the cutoff and filtration used for the knockout cutoff. Additionally, the filtration from Limma is also applied to the data.

Load packages

```
library(ggplot2)
library(reshape2)
library(biomaRt)
library(magrittr)
library(xlsx)
library(mgu74a.db)
library(AnnotationDbi)
library(org.Hs.eg.db)
library(seqinr)
library(grid)
library(gridExtra)
library(edgeR)
library(limma)
library(Ckmeans.1d.dp)
library(pheatmap)
library(RColorBrewer)
```

Functions

The `cutoff()` function is designed to create and above and below cutoff line using the consensus data sets. For every row in the data, it indicates the percentage of genes that are now included in the cutoff point.

```
cutoff <- function(range) {

  above <- table(fullDist[range,"B_D_A_RipDist"]) +
    table(fullDist[range,"B_D_A_ParDist"]) +
    table(fullDist[range,"B_DDist"]) +
    table(fullDist[range,"D_A_ParDist"])


  below <- table(fullDist[length(range):nrow(ncAvg),"B_D_A_RipDist"]) +
    table(fullDist[length(range):nrow(ncAvg),"B_D_A_ParDist"]) +
    table(fullDist[length(range):nrow(ncAvg),"B_DDist"]) +
    table(fullDist[length(range):nrow(ncAvg),"D_A_ParDist"])
```

```
  return((above / (above + below)))
}
```

The annotation function `anno()` is used as a quick way to annotate the columns of the count data returned from Tophat (they are normally in Enterez format).

```
anno <- function(x) {
  require(AnnotationDbi)
  require(org.Mm.eg.db)

  ### Annotate
  x$symbol <- mapIds(org.Mm.eg.db,
                     keys = row.names(x),
                     column = "SYMBOL",
                     keytype = "ENSEMBL",
                     multiVals = "first")

  ### Remove rows with no gene mapping
  x <- subset(x, select = c(17,1:16))
  x <- x[complete.cases(x), ]
  rownames(x) <- make.names(x$symbol, unique = TRUE)
  x[ ,1] <- NULL

  return(x)
}
```

Load data

The count data (filtered for low counts) is loaded here. Replicates are averaged at the saline list is ordered by highest counts.

```
nc <- read.csv("finalCountData.csv", header = TRUE)
rownames(nc) <- nc[ ,1]
nc <- nc[ ,-1]

ncAvg <- data.frame(SAL_WT = rowMeans(nc[ ,1:3]),
                    SAL_KO = rowMeans(nc[ ,4:6]),
                    RO_WT = rowMeans(nc[ ,7:9]),
                    RO_KO = nc[ ,10],
                    RO_RAP_WT = rowMeans(nc[ ,11:13]),
                    RO_RAP_KO = rowMeans(nc[ ,14:16]))

ncAvg <- ncAvg[order(ncAvg$SAL_WT, decreasing = TRUE), ]
```

Determining initial cutoff by comparison to other data

We will first be loading all targets from previous data into R. The Miyashiro and Ascano data will be excluded from the final plotting because it does not have much in common with our list (similar to what was found by *Suhl et al. [1]*). However, we are still providing their data in the code below. The consensus overlap lists from *Suhl et al.* are also included below.

```
### Brown data
brownDat <- read.xlsx("brown_list.xls", sheetIndex = 1, header = TRUE, startRow = 2)
brownProbes <- brownDat[ ,"Probe.Set"] %>% as.vector()
probes <- as.list(mgu74aALIAS2PROBE)
brownGenes <- probes[probes %in% brownProbes] %>% unlist() %>% names()

### Darnell data
darnellDat <- read.xlsx("darnell_list.xls", sheetIndex = 1, header = TRUE, startRow = 2)
darnellGenes <- darnellDat[ ,"Symbol.from.mm9"] %>% as.vector()

### Ascano PAR-CLIP data
ascanoParDat <- read.xlsx("ascano_par.xlsx", sheetIndex = 1, header = TRUE, startRow = 3, colIndex = 1:4
ascanoParGenes <- ascanoParDat[ascanoParDat$Total.mRNA.binding.sites > 0, "Gene"] %>% as.vector()

### Ascano RIP-CHIP data
ascanoRipDat <- read.csv("ascano_rip.csv", header = TRUE, na.strings = "")
ascanoRipDat <- ascanoRipDat[ ,c(1,3)]
ascanoRipDat <- split(ascanoRipDat, ascanoRipDat$Target.bin)
ascanoRipDat <- ascanoRipDat$Target
ascanoRipGenes <- ascanoRipDat[ ,1] %>% as.vector()

### Miyashiro data
miyashiroDat <- read.csv("miyashiro.csv", header = TRUE)
miyashiroDat <- miyashiroDat[-c(1,1153:1159), ]
miyashiroDat <- miyashiroDat[ ,"UniGene"] %>% as.data.frame()
names(miyashiroDat) <- "unigene"
miyashiroDat$unigene <- as.character(miyashiroDat$unigene)
miyashiroDat$symbol <- mapIds(org.Hs.eg.db,
                              keys = miyashiroDat$unigene,
                              column = "SYMBOL",
                              keytype = "UNIGENE",
                              multiVals = "first")
miyashiroGenes <- miyashiroDat$symbol %>% na.omit() %>% as.vector()

### Consensus data
BDARIP <- read.xlsx("consensus_FMRP.xlsx", sheetIndex = 2, header = FALSE, startRow = 2, stringsAsFactor
BDAPAR <- read.xlsx("consensus_FMRP.xlsx", sheetIndex = 3, header = FALSE, startRow = 2, stringsAsFactor
BD <- read.xlsx("consensus_FMRP.xlsx", sheetIndex = 4, header = FALSE, startRow = 2, stringsAsFactors =
DAPAR <- read.xlsx("consensus_FMRP.xlsx", sheetIndex = 5, header = FALSE, startRow = 2, stringsAsFactors
```

Using the above data, we can now construct a large table that includes all the genes in long format for plotting with ggplot2. The code below also has a few extra details to get the factor levels in the correct order.

```
### Distributions across ranked saline genes for full data sets
fullDist <- data.frame(row.names = row.names(ncAvg) %>% toupper(),
                       order = 1:nrow(ncAvg),
                       brownDist = ifelse((row.names(ncAvg) %>% toupper()) %in% (brownGenes %>% toupper
                       darnellDist = ifelse((row.names(ncAvg) %>% toupper()) %in% (darnellGenes %>% tou
                       ascanoParDist = ifelse((row.names(ncAvg) %>% toupper()) %in% (ascanoParGenes %>%
                       ascanoRipDist = ifelse((row.names(ncAvg) %>% toupper()) %in% (ascanoRipGenes %>%
                       miyashiroRipDist = ifelse((row.names(ncAvg) %>% toupper()) %in% (miyashiroGenes
                       B_D_A_RipDist = ifelse((row.names(ncAvg) %>% toupper()) %in% (BDARIP %>% toupper
```

```
                        B_D_A_ParDist = ifelse((row.names(ncAvg) %>% toupper()) %in% (BDAPAR %>% toupper
                        B_DDist = ifelse((row.names(ncAvg) %>% toupper()) %in% (BD %>% toupper()), "B_D"
                        D_A_ParDist = ifelse((row.names(ncAvg) %>% toupper()) %in% (DAPAR %>% toupper())

fullDist.m <- melt(fullDist, id.vars = "order")

fullDist.m$value <- factor(fullDist.m$value, levels = rev(c("Darnell", "Brown", "B_D_A-RIP", "B_D_A-PAR
fullDist.m$variable <- factor(fullDist.m$variable, levels = rev(c("darnellDist", "brownDist", "B_D_A_Ri

fullDist.m <- fullDist.m[complete.cases(fullDist.m), ]
```

Next, we constructed a cutoff line that was able to capture 95% of the genes within the consensus data list. To do this, we use the `cutoff()` function described above. For each gene in the data that is included it reports the overlap percentage with the consesus list once the for loop is run. Note that the for loop could be better optimized as it takes quite a while to run. The final data frame `orderedCoverage` contains all the info as a percentage. Since the loop takes a while to run, the following code block can be skipped and once simple load in `orderedCoverage.csv`.

```
### Above/below cutoff line
coverage = NULL
for(i in 1:nrow(ncAvg)) {
  coverage[i] <- cutoff(1:i)
}

orderedCoverage <- data.frame(order = 1:nrow(ncAvg),
                              coverage = coverage * 100)
```

Finally, the data is filtered by including 95% of the genes that overlap with the other data sets. There are two graphs show below. One shows each gene found in our data with the other data sets and shows the cutoff line (shown in red). The bottom graph then shows the percentage of genes that overlap within each row of our ordered data set using the information from `orderedCoverage()`. The filtered list (4120 genes in total) will now be used moving forward for the next filtration steps.

(figure not rendered, see FigureS3.Rmd)

```
consensusCutoff <- orderedCoverage[orderedCoverage$coverage < 95, ] %>% nrow()

### Consensus percentage graph
a <- ggplot(orderedCoverage, aes(x = order, y = coverage)) +
  geom_line() +
  geom_vline(xintercept = consensusCutoff, color = "red", linetype = "dashed") +
  xlab("Ranked Saline Genes") +
  ylab("Percent Consensus Lists in Cutoff") +
  theme_bw()

### Consensus overlap graph
b <- ggplot(fullDist.m, aes(x = order, y = value)) +
  geom_point(alpha = 0.75, color = "dodgerblue") +
  geom_vline(xintercept = consensusCutoff, color = "red", linetype = "dashed") +
  xlab("") +
  ylab("") +
  theme_bw()
```

```
### Fully arranged graphs
gA <- ggplotGrob(a)
gB <- ggplotGrob(b)
grid::grid.newpage()
grid::grid.draw(rbind(gB, gA))
```

Determining KO cutoff by comparison to other data

We have taken two approaches to removing high background counts. The first approach is show here. We know potential targets and non-targets from previous data (Darnell et al. 2011 [2]). By determining the enrichment ratio between WT and KO for a given treatment, we can see where distributions of enrichment ratios for the targets or non-targets fall for all the consensus data sets. Using this, we can calibrate and approximate acceptable range of background for our cutoff.

```
### Create enrichment ratios of WT over KO for each treatment
ncAvgDif <- data.frame(row.names = row.names(ncAvg),
                       SAL = ncAvg$SAL_WT / ncAvg$SAL_KO,
                       RO = ncAvg$RO_WT / ncAvg$RO_KO,
                       RO_RAP = ncAvg$RO_RAP_WT / ncAvg$RO_RAP_KO)

### Load targets and non-targets
targets <- c("Gabbr1", "Gabbr2", "Nisch", "Prrc2a", "Kif1a", "Grin2a", "Adgrb2", "Cyfip2", "Map1b", "Bs
nontargets <- c("Pabpc1", "Tmem65", "Hprt1", "St8sia3", "Sae1", "Glrb", "Gria2", "Tcerg1", "Eif3a", "Sl

### Create a data frame that includes the overlap of genes with our data set for each consensus gene li
foldRange <- rbind(data.frame(group = "Brown", value = ncAvgDif[row.names(ncAvgDif) %in% brownGenes, ]$
                   data.frame(group = "Darnell", value = ncAvgDif[row.names(ncAvgDif) %in% darnellGenes
                   data.frame(group = "B_D_A-RIP", value = ncAvgDif[(row.names(ncAvgDif) %>% toupper())
                   data.frame(group = "B_D_A-PAR", value = ncAvgDif[(row.names(ncAvgDif) %>% toupper())
                   data.frame(group = "B_D", value = ncAvgDif[(row.names(ncAvgDif) %>% toupper()) %in% I
                   data.frame(group = "D_A-PAR", value = ncAvgDif[(row.names(ncAvgDif) %>% toupper()) %
                   data.frame(group = "Non-Targets", value = ncAvgDif[row.names(ncAvgDif) %in% nontarge

### Determine average enrichment ratio range between WT and KO for all groups
foldRange.s <- split(foldRange, foldRange$group)

### Take the mean of the lower quartile of the ranges for all groups (indicated by the dotted line)
cutoff <- sapply(foldRange.s, function(x) quantile(x$value, prob = 0.25))[1:6] %>% mean()
```

This gives us a cutoff of `1.112788`. In other words, a fold enrichment of this value is believed to be the acceptable threshold for where background counts occur.

Determining KO cutoff by univariate K-means

We sought to find a more unbiased, quantitative means of assigning a threshold to the background counts. To do this, we used univariate K-means to attempt to separate out a pattern of background WT/KO enrichment from signal from the filtered data set (4120 genes).

```
ncAvgCut <- ncAvg[1:consensusCutoff, ]
ncAvgCut$"Saline" <- ncAvg[1:consensusCutoff, "SAL_WT"] / ncAvg[1:consensusCutoff, "SAL_KO"]
ncAvgCut$"Ro" <- ncAvg[1:consensusCutoff, "RO_WT"] / ncAvg[1:consensusCutoff, "RO_KO"]
```

```
ncAvgCut$"Ro+Rapa" <- ncAvg[1:consensusCutoff, "RO_RAP_WT"] / ncAvg[1:consensusCutoff, "RO_RAP_KO"]

### Univariate k-means clustering. Arrange output in data frame
clustDat <- data.frame(row.names = row.names(ncAvgCut),
                       enriched = ncAvgCut$Saline,
                       cluster = Ckmeans.1d.dp(ncAvgCut$Saline, 2)$cluster)

### Define a second cutoff by taking the max in the "background" cluster
cutoff2 <- split(clustDat, clustDat$cluster)[[1]]$enriched %>% max()
cutoff2
```

WT/KO enrichment ratio with cutoff established

We are now in a position to examine the final KO cutoff using our two separate methods. Here we simply take the average of the two cutoffs for a final cutoff assigned in the variable `KOcutoff`. This cutoff will be applied to the data by applying it to the saline fraction only. The resulting gene list will be our filtered targets.

```
### Define knockout cutoff
KOcutoff <- mean(cutoff, cutoff2)
KOcutoff

### Filter data to obtain this cutoff
filteredTargets <- ncAvgCut[ncAvgCut$Saline > KOcutoff, 1:6]
```

Filtering mitochondrial and glial cells

To remove mitchondrial and glial cells, we searched the MitoMiner [3] database and the Brain RNA-seq database for cell types [4]

```
mito <- read.csv("mito.csv", header = FALSE, stringsAsFactors = FALSE)$V1
glia <- read.csv("glial.csv", header = FALSE, stringsAsFactors = FALSE)$V1

filteredTargets <- filteredTargets[!(row.names(filteredTargets) %in% c(mito, glia)), ]

finalCountData <- nc[row.names(nc) %in% row.names(filteredTargets), ]
```

Final filtration with limma

Finally, we take our filtered list and apply **limma+voom** for differential expression analysis.

```
### Preprocess data columns
countData <- read.csv(file = "combinedCounts.csv", header = TRUE, stringsAsFactors = FALSE)
rownames(countData) <- countData[ ,1]
countData[ ,1] <- NULL
cdMain <- subset(countData, select = c(7:12, 16:18, 3:6, 13:15)) %>% anno()
cdMain <- cdMain[ ,1:9]
cdMain <- cdMain[row.names(cdMain) %in% row.names(finalCountData), ]

### Analyze with limma+voom
dge <- DGEList(counts = cdMain)
```

```
dge <- calcNormFactors(dge)
dge <- cpm(dge, log = TRUE, prior.count = 3)
design <- model.matrix(~ 0+factor(c(1,1,1,2,2,2,3,3,3)))
colnames(design) <- c("RO_RAP_WT", "RO_WT", "SAL_WT")

contrast.matrix <- makeContrasts(SAL_WT-RO_WT, RO_RAP_WT-RO_WT, levels = design)

v <- voom(cdMain, design, plot = FALSE)
fit <- lmFit(v, design)
fit2 <- contrasts.fit(fit, contrast.matrix)
fit2 <- eBayes(fit2)
resultsRo <- topTable(fit2, coef = 1, number = 25000)
resultsRoRap <- topTable(fit2, coef = 2, number = 25000)

resultsRo$Threshold <- ifelse(resultsRo$adj.P.Val < 0.1, "Significant", "Non-Significant")
resultsRoRap$Threshold <- ifelse(resultsRo$adj.P.Val < 0.1, "Significant", "Non-Significant")
```

References

[1] Suhl, J.A, Chopra, P., Anderson, B.R., Bassell, G.J., Warren, S.T. (2014). Analysis of FMRP mRNA target datasets reveals highly associated mRNAs mediated by G-quadruplex structures formed via clustered WGGA sequences. *Hum Mol Genet* 23, 5479-5491.

[2] Darnell, J.C., Van Driesche, S.J., Zhang, C., Hung, K.Y., Mele, A., Fraser, C.E., Stone, E.F., Chen, C., Fak, J.J., Chi, S.W., Licatalosi, D.D., Richter, J.D., Darnell, R.B. (2011). FMRP stalls ribosomal translocation on mRNAs linked to synaptic function and autism. *Cell* 146, 247-261.

[3] Smith, A.C., Blackshaw, J.A., Robinson, A.J. (2012). MitoMiner: a data warehouse for mitochondrial proteomics data. *Nucleic Acids Res* 40(Database issue), D1160-D1167.

[4] Zhang, Y., Chen, K., Sloan, S.A., Bennett, M.L., Scholze, A.R., O'Keeffe, S., Phatnani, H.P., Guarnieri, P., Caneda, C., Ruderisch, N., Deng, S., Liddelow, S.A., Zhang, C., Daneman R., Maniatis, T., Barres, B.A., Wu, J.Q. (2014). An RNA-Sequencing Transcriptome and Splicing Database of Glia, Neurons, and Vascular Cells of the Cerebral Cortex. *J Neurosci* 34(36), 11929-11947.

Session info: