LabBook_13_03_2017

Claire Green

Monday

I need to work out STILL a way of assigning the threshold to use for my DEG overlap analysis. This has been plaguing me for a long time and I really don't know exactly how I'm going to tackle it. The spike in method that I reported previously works really well, the problem is the fact I can't enrich for disease genes (because they are in the benchmarking list) but also that I CAN'T GET THE SAME RESULTS AGAIN. I even went to the trouble of copying and pasting the script as it was to make sure I used the exact same one, but for some reason something has changed and now it's saying for the same analysis that the optimum threshold is in the 7000s

Here are the methods I have tried so far with the positives and negatives:

1. Assigned threshold
   This was my first method - just take a top threshold of N genes and see what overlaps. This gave relatively good results but the problem was that I knew there was no evidence behind the threshold I had chosen. I thought - there must be a threshold at which my enrichment signal is clearest. Which led me to method number two…

2. Enrichment spike in
   This involved making a meta-list of my benchmarks. This was then used to run enrichment analysis at different thresholds so as to define the exact threshold at which enrichment was highest. This was at first 5996. The positives were that the enrichment of the processes on EnrichR were vastly improved, as were, obviously, the benchmark enrichments. However there were two big problems: 1, I couldn't repeat the 5996 results (even runninhg exaclty the same script gave me a result of 7000 or so) and 2, If I used the benchmarks as the spike-in list then I couln't use it as a bnechmark list - it would be a circular argument. I attempted to cut down the spike-in list to just TDP-43 PPI genes but the problem then was that any functional enrichment analysis I did would be biased towards TDP-43 interactors which enrich heavily for mRNA processing. Therefore any mRNA processing enrichment is skewed towards the thresholding. I did try the enrichment without the overlapping genes which did still (sort of) enrich for mRNA processing but it was far weaker. This was a good back up if the following didn't work.

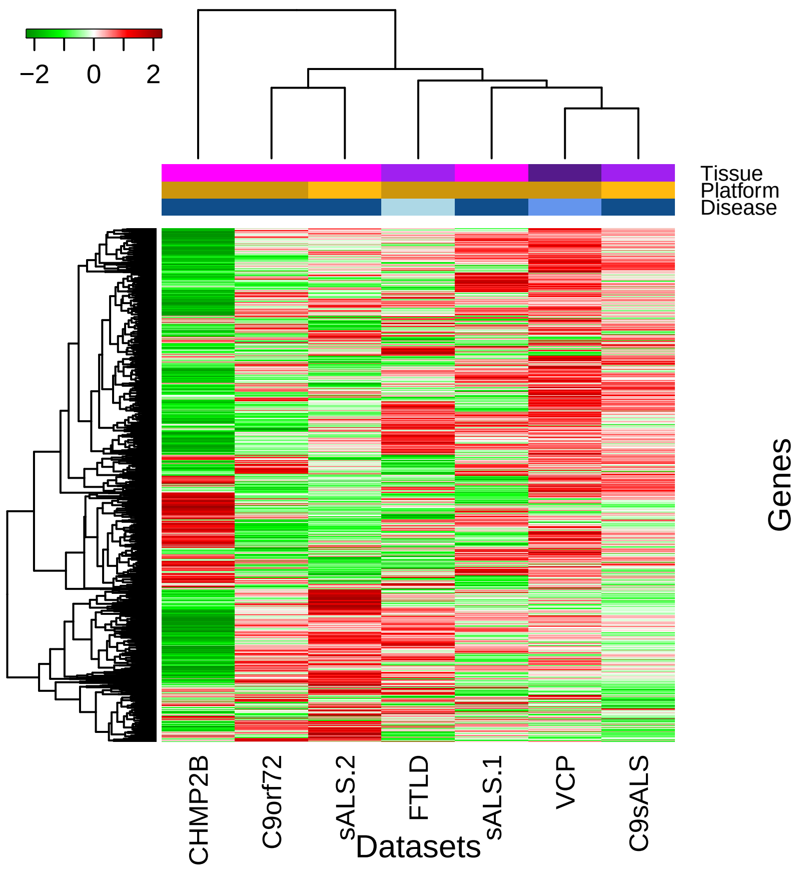
3. Heatmap
   Dennis suggested that a way of identifying up and downregulated genes could be to use the dendrogram of a heatmap to cluster the similarly expressed genes together.

```r
setwd("/users/clairegreen/Documents/PhD/TDP-43/TDP-43_Code/Results/GeneExpression/noMedian/")

C9 <- read.csv("C9_unique.csv")
CH <- read.csv("CH_unique.csv")
sals <- read.csv("sals_unique.csv")
ftld <- read.csv("ftld_unique.csv")
vcp <- read.csv("vcp_unique.csv")

setwd("/users/clairegreen/Documents/PhD/TDP-43/TDP-43_Code/Results/GeneExpression/TDP-43_DEseq2/")

pet <- read.csv("PET_results_keepfiltering.csv")
pet <- pet[!duplicated(pet$hgnc_symbol),]
rav <- read.csv("RAV_results_keepfiltering.csv")
rav <- rav[!duplicated(rav$hgnc_symbol),]
```

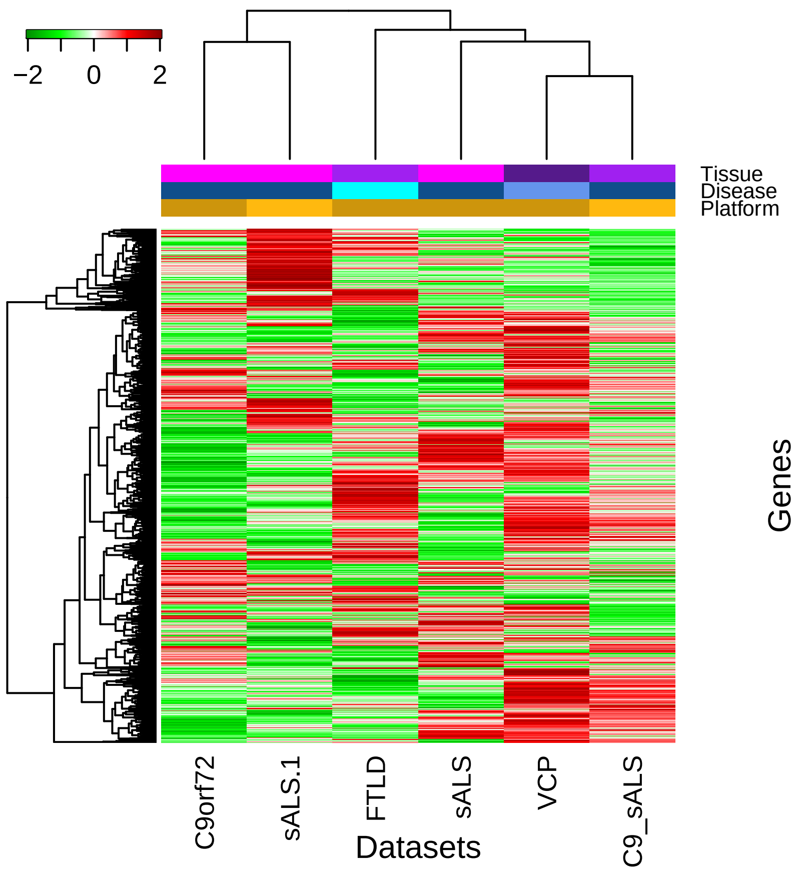
```r
#Find genes that are present across all FULL datasets
C9gen <- C9$Gene.Symbol
CHgen <- CH$Gene.Symbol
salsgen <- sals$Gene.Symbol
ftldgen <- ftld$Gene.Symbol
vcpgen <- vcp$Gene.Symbol
petgen <- pet$hgnc_symbol
ravgen <- rav$hgnc_symbol

genelist <- Reduce(intersect, list(C9gen, CHgen, salsgen, ftldgen, vcpgen, petgen, ravgen))

#take that subset from each of the datasets
subsetC9 <- subset(C9, C9$Gene.Symbol %in% genelist, drop = TRUE)
rownames(subsetC9) <- subsetC9$Gene.Symbol
subsetC9[,1] = NULL
subsetC9 <- subsetC9[order(row.names(subsetC9)),]

subsetCH <- subset(CH, CH$Gene.Symbol %in% genelist, drop = TRUE)
rownames(subsetCH) <- subsetCH$Gene.Symbol
subsetCH[,1] = NULL
subsetCH <- subsetCH[order(row.names(subsetCH)),]

subsetsals <- subset(sals, sals$Gene.Symbol %in% genelist, drop = TRUE)
rownames(subsetsals) <- subsetsals$Gene.Symbol
subsetsals[,1] = NULL
subsetsals <- subsetsals[order(row.names(subsetsals)),]

subsetftld <- subset(ftld, ftld$Gene.Symbol %in% genelist, drop = TRUE)
rownames(subsetftld) <- subsetftld$Gene.Symbol
subsetftld[,1] = NULL
subsetftld <- subsetftld[order(row.names(subsetftld)),]

subsetvcp <- subset(vcp, vcp$Gene.Symbol %in% genelist, drop = TRUE)
rownames(subsetvcp) <- subsetvcp$Gene.Symbol
subsetvcp[,1] = NULL
subsetvcp <- subsetvcp[order(row.names(subsetvcp)),]

subsetpet <- subset(pet, pet$hgnc_symbol %in% genelist, drop = TRUE)
rownames(subsetpet) <- subsetpet$Gene.Symbol
subsetpet[,1] = NULL
subsetpet <- subsetpet[order(subsetpet$hgnc_symbol),]

subsetrav <- subset(rav, rav$hgnc_symbol %in% genelist, drop = TRUE)
rownames(subsetrav) <- subsetrav$Gene.Symbol
subsetrav[,1] = NULL
subsetrav <- subsetrav[order(subsetrav$hgnc_symbol),]


#Generate a matrix with gene names and log fold change values from each
LFC <- data.frame(gene=row.names(subsetC9),
                C9orf72=subsetC9$logFC,
                CHMP2B=subsetCH$logFC,
                sALS.1=subsetsals$logFC,
                FTLD=subsetftld$logFC,
                VCP=subsetvcp$logFC,
                C9sALS=subsetpet$log2FoldChange,
                sALS.2=subsetrav$log2FoldChange)
rownames(LFC) <- LFC$gene
LFC[,1] <- NULL
LFC <- as.matrix(LFC)

LFCNC <- data.frame(gene=row.names(subsetC9),
                   C9orf72=subsetC9$logFC,
                   sALS=subsetsals$logFC,
                   FTLD=subsetftld$logFC,
                   VCP=subsetvcp$logFC,
                   C9_sALS=subsetpet$log2FoldChange,
                   sALS=subsetrav$log2FoldChange)

rownames(LFCNC) <- LFCNC$gene
LFCNC[,1] <- NULL
LFCNC <- as.matrix(LFCNC)

library(gplots)
```

```
##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(heatmap3)
hmcols<- colorRampPalette(c("green4","green","white", "red","red4"))(256)
#
# Platform <- c("orange","orange","orange","orange","blue", "blue")
# Tissue <- c("magenta", "magenta", "yellow", "cyan", "yellow", "magenta")

colsidecolorsLFCNC <- cbind(Platform=c("darkgoldenrod3","darkgoldenrod3","darkgoldenrod3","darkgoldenro
                       Disease=c("dodgerblue4","dodgerblue4","cyan","cornflowerblue","dodgerblue4","dod
                       Tissue=c("magenta", "magenta", "purple", "purple4", "purple", "magenta"))

colsidecolorsLFC <- cbind(Disease=c("dodgerblue4","dodgerblue4","dodgerblue4","lightblue","cornflowerbl
                          Platform=c("darkgoldenrod3","darkgoldenrod3","darkgoldenrod3","darkgoldenrod3
                          Tissue=c("magenta","magenta", "magenta", "purple", "purple4", "purple", "mage

heatmap3(LFC,
         col = hmcols,
         Rowv = TRUE,
         Colv = TRUE,
         distfun = dist,
         hclustfun = hclust,
         scale = "row",
         labCol = colnames(LFC),
         ColSideColors = colsidecolorsLFC,
         ColSideWidth = 1,
         cexCol = 1.2,
         cexRow = 0.01,
         xlab = "Datasets",
         ylab = "Genes")
```



```
heatmap3(LFCNC,
         col = hmcols,
         Rowv = TRUE,
         Colv = TRUE,
         distfun = dist,
         hclustfun = hclust,
         scale = "row",
         labCol = colnames(LFCNC),
         ColSideColors = colsidecolorsLFCNC,
         ColSideWidth = 1,
         cexCol = 1.2,
         cexRow = 0.01,
         xlab = "Datasets",
         ylab = "Genes")
```



As you can see from the first heatmap, the CHMP2B dataset has really odd expression in comparison to the others. The colours are very bright and almost opposite to every other sample. At this point although I decided the heatmap wouldn't necessarily be the best option, I don't think including the CHMP2B dataset any further is a good idea. This will change a lot of results…

3. Fold change
   My thought then was to try and use a measure that was completely unrelated to the benchamrk list. I wanted to avoid p value as it's so scrutinised and arbitrary so I thought I would try fold change instead. On Friday I began these tets but the problem was that for log2FC I needed such a low threshold that I thought it looked a bit dodgy. I'm not really familiar with the field but using a threshold of 0.1 felt not so great. I don't know if this really makes a difference but I decided to use fold change adjusted to a scale centered around 0.

```r
setwd("/users/clairegreen/Documents/PhD/TDP-43/TDP-43_Code/Results/GeneExpression/noMedian/")

C9 <- read.csv("C9_unique.csv")
C9 <- C9[order(C9$P.Value),]
CH <- read.csv("CH_unique.csv")
CH <- CH[order(CH$P.Value),]
sals <- read.csv("sals_unique.csv")
sals <- sals[order(sals$P.Value),]
ftld <- read.csv("ftld_unique.csv")
ftld <- ftld[order(ftld$P.Value),]
vcp <- read.csv("vcp_unique.csv")
vcp <- vcp[order(vcp$P.Value),]

setwd("/users/clairegreen/Documents/PhD/TDP-43/TDP-43_Code/Results/GeneExpression/TDP-43_DEseq2/")

pet <- read.csv("PET_results_keepfiltering.csv")
pet <- pet[!duplicated(pet$hgnc_symbol),]
pet$"FoldChange"<-2^pet$log2FoldChange
pet$"FoldChange"[pet$"FoldChange"<1]<-(-1)/pet$"FoldChange"[pet$"FoldChange"<1]
rav <- read.csv("RAV_results_keepfiltering.csv")
rav <- rav[!duplicated(rav$hgnc_symbol),]
rav$"FoldChange"<-2^rav$log2FoldChange
rav$"FoldChange"[rav$"FoldChange"<1]<-(-1)/rav$"FoldChange"[rav$"FoldChange"<1]

thresh <- 1

upC9 <- subset(C9, C9$Fold.Change >= thresh)
upC9gene <- upC9$Gene.Symbol

# upCH <- subset(CH, CH$Fold.Change >= thresh)
# upCHgene <- upCH$Gene.Symbol

upSALS <- subset(sals, sals$Fold.Change >= thresh)
upSALSgene <- upSALS$Gene.Symbol

upFTLD <- subset(ftld, ftld$Fold.Change >= thresh)
upFTLDgene <- upFTLD$Gene.Symbol

upVCP <- subset(vcp, vcp$Fold.Change >= thresh)
upVCPgene <- upVCP$Gene.Symbol

upPET <- subset(pet, pet$FoldChange >= thresh)
upPETgene <- upPET$hgnc_symbol

upRAV <- subset(rav, rav$FoldChange >= thresh)
upRAVgene <- upRAV$hgnc_symbol
```

```r
INTUP <- Reduce(intersect, list(upC9gene, upSALSgene, upFTLDgene, upVCPgene, upPETgene, upRAVgene))

cat(INTUP, sep = "\n")

setwd("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Code/Results/GeneExpression/FoldChangeResults")
write.table(INTUP, "intersect_up_1.txt", col.names = F, row.names = F, quote = F)



#### DOWN ####
thresh <- -1

downC9 <- subset(C9, C9$Fold.Change <= thresh)
downC9gene <- downC9$Gene.Symbol

# downCH <- subset(CH, CH$Fold.Change <= thresh)
# downCHgene <- downCH$Gene.Symbol

downSALS <- subset(sals, sals$Fold.Change <= thresh)
downSALSgene <- downSALS$Gene.Symbol

downFTLD <- subset(ftld, ftld$Fold.Change <= thresh)
downFTLDgene <- downFTLD$Gene.Symbol

downVCP <- subset(vcp, vcp$Fold.Change <= thresh)
downVCPgene <- downVCP$Gene.Symbol

downPET <- subset(pet, pet$FoldChange <= thresh)
downPETgene <- downPET$hgnc_symbol

downRAV <- subset(rav, rav$FoldChange <= thresh)
downRAVgene <- downRAV$hgnc_symbol

INTDOWN <- Reduce(intersect, list(downC9gene, downSALSgene, downFTLDgene, downVCPgene, downPETgene, dow

setwd("/Users/clairegreen/Documents/PhD/TDP-43/TDP-43_Code/Results/GeneExpression/FoldChangeResults")
write.table(INTDOWN, "intersect_down_1.txt", col.names = F, row.names = F, quote = F)

cat(INTDOWN, sep = "\n")
```

As you can see, I decided to remove the CHMP2B dataset. Taking upregulated genes with FC above or equal to 1, and downregulated genes as below or equal to -1, I ended up with 329 upregulated genes and 69 downregulated genes.

Upregulated Genes

1-30	31-60	61-90	91-120	121-15	151-180	181-210	211-240	241-270
RALGDS	CTBP2	PIKFYVE	ZNF573	ABCA8	TTF1	AGO3	SETD2	NBR1
AHCYL1	CALD1	GPNMB	HIPK1	TMED10	WDR11	ALMS1	CREG1	UNC13B
EEF1A1	PAPOLA	PABPC1	ZNF12	HSBP1	MYOF	RANGRF	CEP57	FAM65B
RCN1	NOTCH2	SMC5	RND3	PPFIA1	SRRM1	YY1AP1	RPRD1A	MARK3
STOM	SLCO3A1	VCAN	PRPF40A	AASS	CLK4	GIN1	ZFX	MAP4K5
IFITM2	LPL	PHLDA1	SNAPC3	TRIT1	ADAMTS9	ABL1	HNRNPA2B1	PPP1R12B
KCTD20	CCNL1	CCNL2	ITPR2	GNA13	KDM5A	PTK2	LIMA1	SH3GLB1
CTSH	SMAD5	AMD1	LPP	SPEN	THUMPD2	PIGA	MON2	PLEKHA2
MAFF	ATG14	GSPT1	CDH11	CMTM6	PIK3C2A	LBR	ZCCHC8	EIF3E
LYN	ZFYVE26	EPS8	ZCCHC6	SRSF4	ANKRD10	SLK	UPF3B	UBXN7
PRPF4B	PPP4R2	N4BP2L2	PNN	TARDBP	PRR11	SUCO	TIMM8A	CLIC4
CNOT6	SASH1	TBC1D2B	AUTS2	KATNBL1	ABI1	MARCH3	SORBS1	ZCCHC10
SGK1	ZBTB10	NSMAF	PUM2	HCFC2	RSL24D1	STK3	GSAP	SLC26A2
EDNRB	MLH3	KCTD12	CRISPLD2	ICE1	TMX1	SF3A1	NBPF1	USP3
KAT6A	NUDT21	TMCO1	TIMP2	DERL1	ADAMTS1	ANKRD28	TENM1	C6orf120
SPARC	DDX21	NKTR	ESF1	PCSK5	USP1	TNFAIP3	KRIT1	KIAA0141
RBM6	HNRNPA3	NR2C1	RPS6KA2	NUP43	ERBB4	PXDN	SREK1	MED14
ASAH1	PSAP	ROCK1	YTHDF1	PRPSAP1	PTPRA	JMJD1C	WTAP	CPOX
RBPJ	ALOX5	RNASEH1	GOLGA7	SCAF11	MAP4K3	CRLF3	NEK1	ANXA1
ATF7IP	BTG1	CBFB	SEC24A	GNAI3	SLC11A2	ZFHX3	REL	WSB1
CRY1	IFI16	MCMBP	PALLD	CHSY1	RAP1B	CDK13	STRN3	FHL1
PHIP	CREB3L2	MFN1	PKIA	RAB27A	IFRD1	ZNF83	SLC33A1	SH2B3
FAM129A	TRAK1	ZNF146	SMAD1	ARMC1	SSFA2	PHF21A	SOCS2	HEXB
DDX18	KCNE4	LRBA	CPSF7	PDE4B	HMGN4	PROSER1	ZNF536	TCF12
PLOD2	PHF3	COL4A1	ZBTB5	CARD8	PDS5A	PTPRC	POGZ	TPP1
SERBP1	PTP4A3	ZNF292	RAP1A	HLA-B	KLF6	MAN2A1	AKR1B1	TRIM38
TMEM43	HCLS1	SOAT1	STX16	GCH1	RBBP6	ITGA6	DIP2A	OGT
SEPT8	PCNA	LAPTM5	EFEMP1	ZFC3H1	RBM5	ADNP	PLSCR1	MUS81
MED13L	CNOT1	AMPD3	SAT1	ALPK1	SUN1	TWSG1	U2SURP	CHD1
ADAM17	NEK4	PNISR	CEBPZ	CDK12	ANAPC5	HEATR3	ASXL2	RIT1

Downregulated genes

1-30	31-60	61-69
COX5A	PEF1	MARK4
DRG1	HABP4	MED8
SLC41A3	WIPI2	CXorf56
SMARCA2	KCNMB4	DEDD
VDAC1	SAMM50	CLCN6
PHYH	MGST3	PSMF1
PCDH17	TPGS2	LINC00094
BNIP3	CDH22	TCEAL1
RIMS1	SPOCK1	CYP2E1
APLP2	TSPYL5	
VDAC3	GDI1	
PPP1R11	FXR2	
SLC12A5	SEPT6	
RPH3A	LRRC14	
IMPDH1	ALDOA	

1-30	31-60	61-69
GNB5	UCHL1	
WASF1	EIF1B	
PSMB2	SIGMAR1	
ESRRG	PMS2P1	
TMX4	SSBP3	
ZBTB7A	TMOD2	
PLCB1	DOLPP1	
PPP3CA	CEP170B	
C16orf62	R3HCC1	
KCNAB1	SNX11	
AUH	GIPC1	
OAT	PDE4A	
CDIPT	ANKRD27	
NDEL1	TXN2	
CLCN3	LPCAT4	