nature portfolio

https://doi.org/10.1038/s41564-021-00882-3

Supplementary information

Detection of respiratory syncytial virus defective genomes in nasal secretions is associated with distinct clinical outcomes

In the format provided by the authors and unedited

Host transcriptome analysis of 13 hospitalized pediatric patients

Yan Sun

3/17/2020

Step 1: Sleuth_TxImport

```
library(tidyverse) # provides access to Hadley Wickham's collection of R packages for data science, whi
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tximport) # package for getting Kallisto results into R
library(biomaRt) # provides access to a wealth of annotation info
targets <- read.table("studyDesign_h.txt", row.names=NULL, header = T, as.is = T)# read in your study d
path <- file.path(targets$sample, "abundance.h5")
# now check to make sure this path is correct by seeing if the files exist
all(file.exists(path))
```

```
## [1] TRUE
```

```
targets <- mutate(targets, path) # add paths to your study design (only necessary for Sleuth)
Hs.anno <- useMart(biomart="ENSEMBL_MART_ENSEMBL", dataset = "hsapiens_gene_ensembl") # select 'mart' f
Tx <- getBM(attributes=c('ensembl_transcript_id_version', # get gene symbols for each transcript ID
                         'external_gene_name'),
            mart = Hs.anno)
Tx <- as_tibble(Tx) # convert this annotation mapping file to a tibble (the tidyverse version of a data
Txi_gene <- tximport(path, #reading kallisto data into R
                     type = "kallisto",
                     tx2gene = Tx,
                     txOut = FALSE,
                     countsFromAbundance = "lengthScaledTPM")
```

```
## 1
```

```
## 2 3 4 5 6 7 8 9 10 11 12 13
## transcripts missing from tx2gene: 4823
## summarizing abundance
## summarizing counts
## summarizing length
## summarizing inferential replicates
```

```
myCPM <- as_tibble(Txi_gene$abundance, rownames = "geneSymbol") # these are you counts after adjusting
```

```
## Warning: `as_tibble.matrix()` requires a matrix with column names or a `.name_repair` argument. Using
## This warning is displayed once per session.
```

```
myCounts <- as_tibble(Txi_gene$counts, rownames = "geneSymbol") # these are your transcript per million
```

Step 2: dataWrangling

```
library(RColorBrewer)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(genefilter)
```

```
##
## Attaching package: 'genefilter'
```

```
## The following object is masked from 'package:readr':
##
##     spec
```

```
library(edgeR)
```

```
## Loading required package: limma
```

```
library(matrixStats)
```

```
##
## Attaching package: 'matrixStats'
```

```
## The following objects are masked from 'package:genefilter':
##
##     rowSds, rowVars
```

```
## The following object is masked from 'package:dplyr':
##
##     count
```

```
groups1 <- targets$DVGgroup
groups1 <- factor(groups1)
groups2 <- targets$Severity
groups2 <- factor(groups2)
groups3 <- targets$DVG.Severity
groups3 <- factor(groups3)
sampleLabels <- targets$sample

myDGEList <- DGEList(Txi_gene$counts)
save(myDGEList, file = "myDGEList_all")
load(file = "myDGEList_all")
log2.cpm <- cpm(myDGEList, log=TRUE)
```

```
nsamples <- ncol(log2.cpm)
myColors <- brewer.pal(nsamples, "Paired")
```

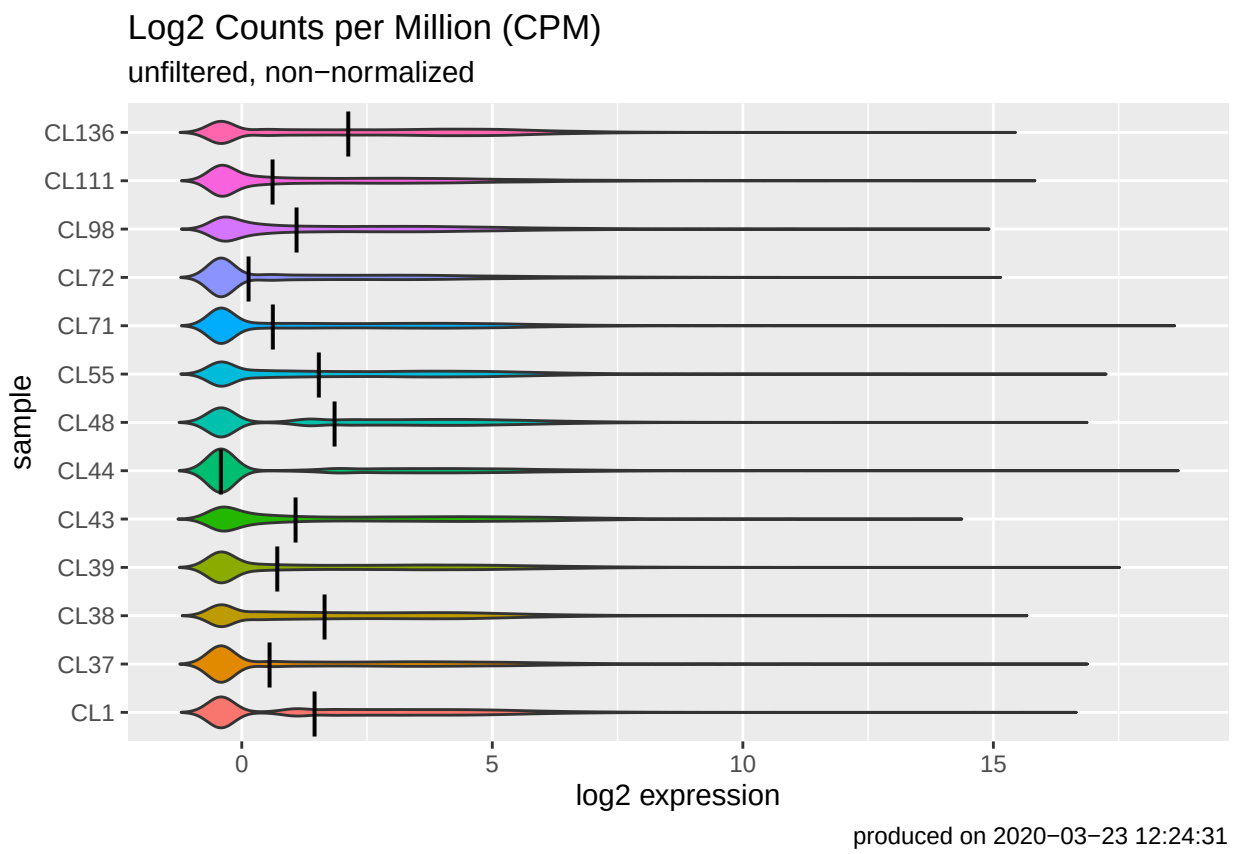
```
## Warning in brewer.pal(nsamples, "Paired"): n too large, allowed maximum for palette Paired is 12
## Returning the palette you asked for with that many colors
```

```
log2.cpm.df <- as_tibble(log2.cpm)
colnames(log2.cpm.df) <- sampleLabels
log2.cpm.df.melt <- melt(log2.cpm.df)
```

```
## No id variables; using all as measure variables
```

```
ggplot(log2.cpm.df.melt, aes(x=variable, y=value, fill=variable)) +
  geom_violin(trim = FALSE, show.legend = FALSE) +
  stat_summary(fun.y = "median", geom = "point", shape = 124, size = 6, color = "black", show.legend = F
  labs(y="log2 expression", x = "sample",
       title="Log2 Counts per Million (CPM)",
       subtitle="unfiltered, non-normalized",
       caption=paste0("produced on ", Sys.time())) +
  coord_flip()
```

```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```



Step3: filter the low expression genes

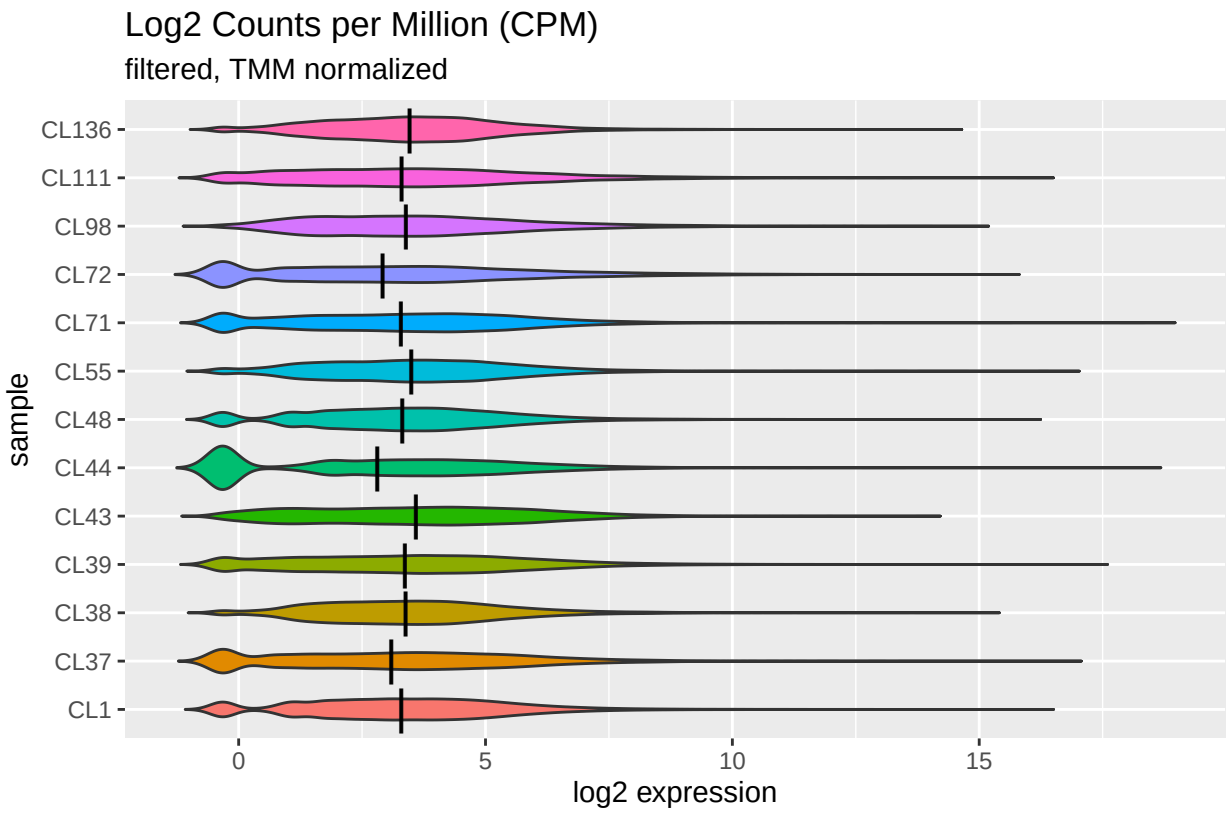
```
keepers <- rowSums(log2.cpm>1)>=4
myDGEList.filtered <- myDGEList[keepers,]
myDGEList.filtered.norm <- calcNormFactors(myDGEList.filtered, method = "TMM")
```

```
log2.cpm.filtered.norm <- cpm(myDGEList.filtered.norm, log=TRUE)
log2.cpm.filtered.norm.df <- as_tibble(log2.cpm.filtered.norm)
colnames(log2.cpm.filtered.norm.df) <- sampleLabels
log2.cpm.filtered.norm.df.melt <- melt(log2.cpm.filtered.norm.df)
```

```
## No id variables; using all as measure variables
```

```
ggplot(log2.cpm.filtered.norm.df.melt, aes(x=variable, y=value, fill=variable)) +
  geom_violin(trim = FALSE, show.legend = FALSE) +
  stat_summary(fun.y = "median", geom = "point", shape = 124, size = 6, color = "black", show.legend = F
  labs(y="log2 expression", x = "sample",
       title="Log2 Counts per Million (CPM)",
       subtitle="filtered, TMM normalized",
       caption=paste0("produced on ", Sys.time())) +
  coord_flip()
```

```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```



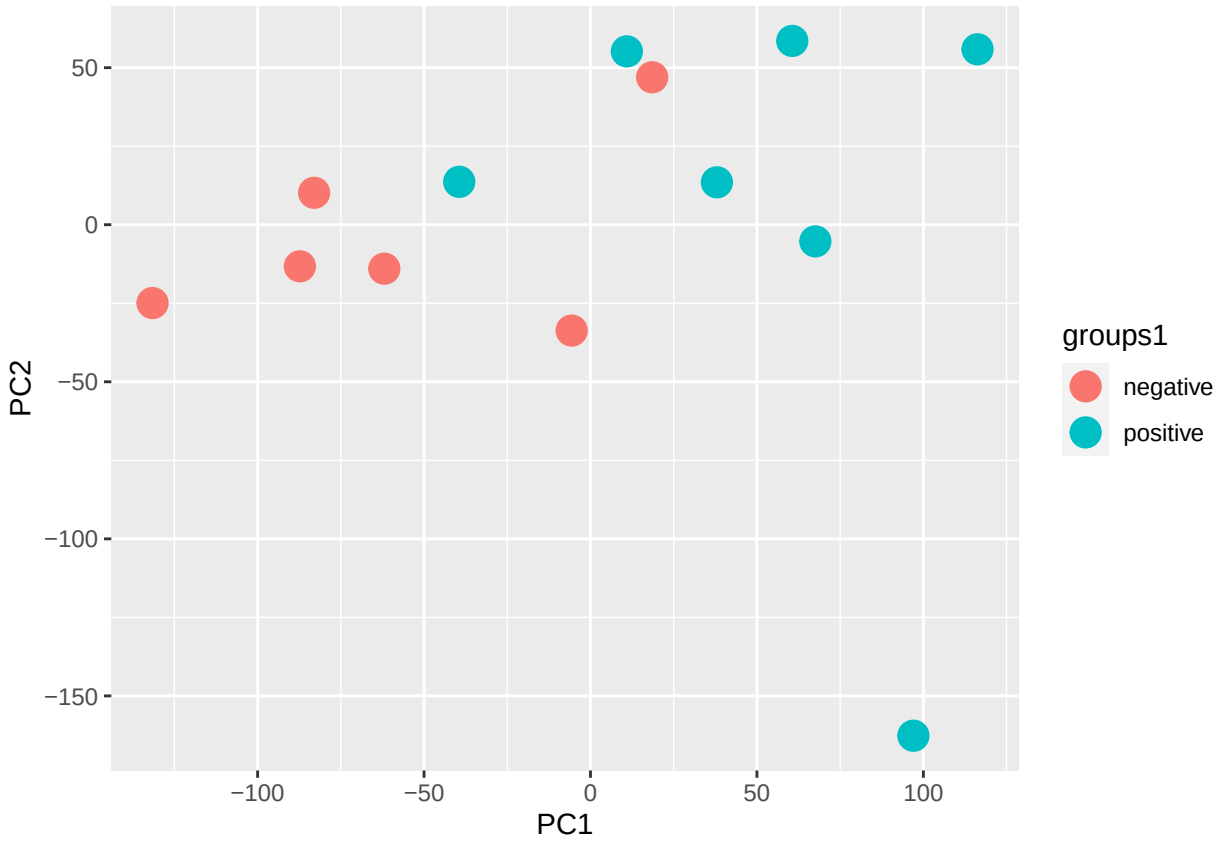
Step 4: multivariate

```
library(tidyverse)
library(reshape2)
library(DT)
library(gt)
library(plotly)
```

```
##
```

```
## Attaching package: 'plotly'

## The following object is masked from 'package:biomaRt':
##
##     select

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```
pca.res <- prcomp(t(log2.cpm.filtered.norm), scale.=F, retx=T)
pc.var<-pca.res$sdev^2
pc.per<-round(pc.var/sum(pc.var)*100, 1)

pca.res.df <- as_tibble(pca.res$x)
ggplot(pca.res.df, aes(x=PC1, y=PC2, color=groups1)) +
  geom_point(size=5) +
  theme(legend.position="right")
```

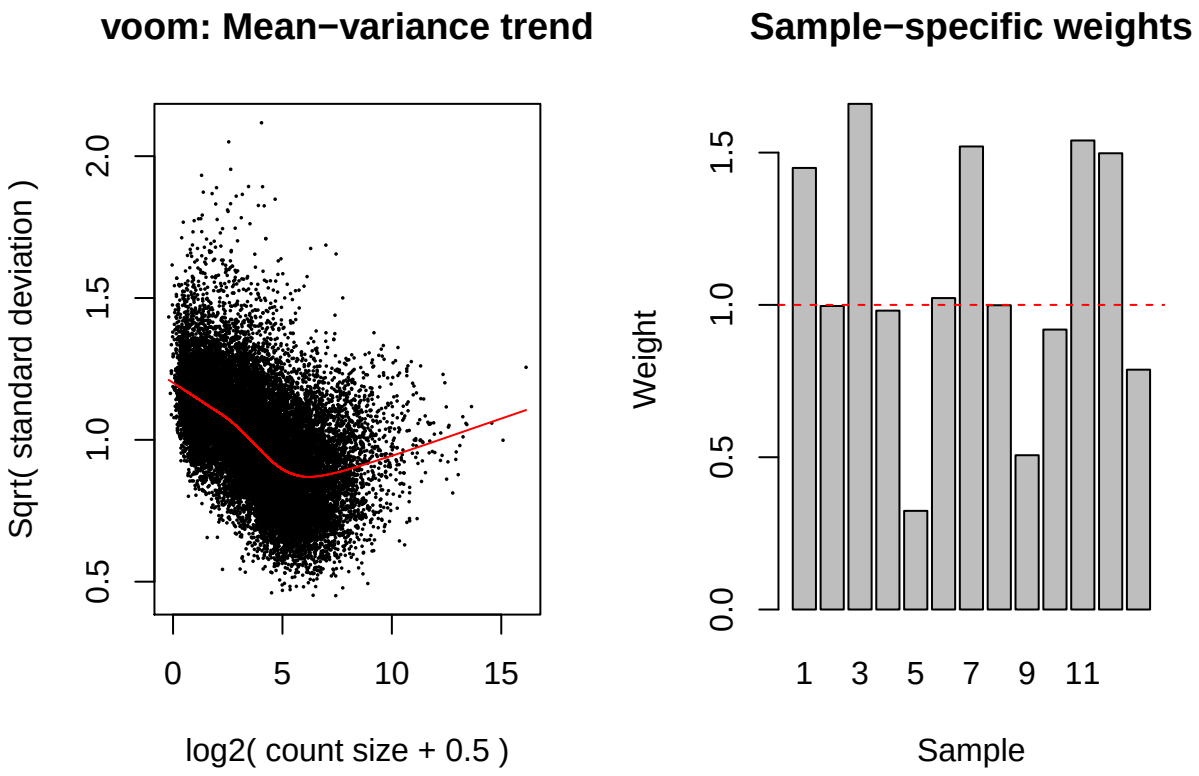


Step 6: diffGenes

```
library(tidyverse)
library(limma) #powerful package for differential gene expression using linear modeling
library(edgeR) #another great package for differential gene expression analysis
library(gt)
library(DT) #creates interactive datatables
library(plotly)
library(statmod)

design <- model.matrix(~0 + groups1)
colnames(design) <- levels(groups1)
rownames(design) <- targets$sample
corfit <- duplicateCorrelation(myDGEList.filtered.norm$counts, design, block=targets$sample)
corfit$consensus
```

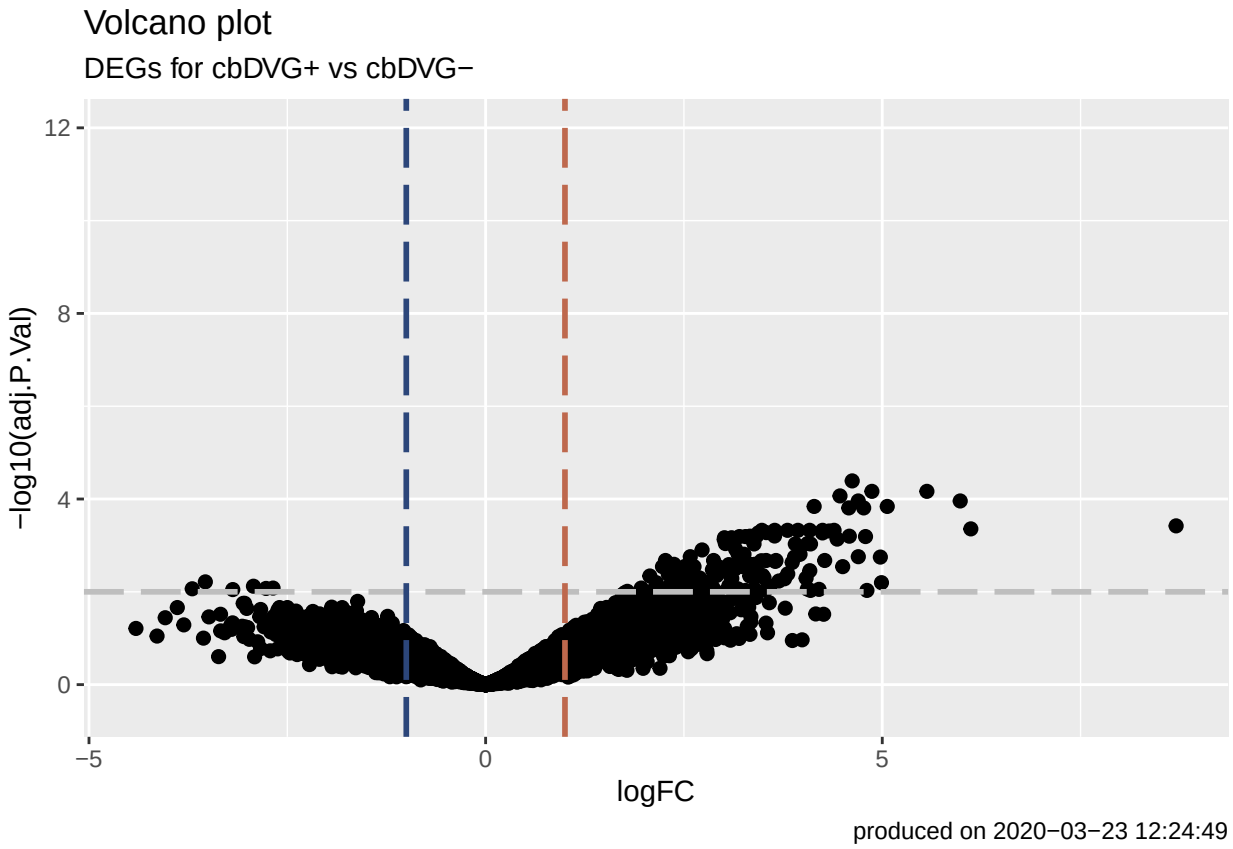
```
## [1] NaN
```

```
v.DEGList.filtered.norm <- voomWithQualityWeights(myDGEList.filtered.norm, design, plot = TRUE)
```




```
#v.DEGList.filtered.norm <- voom(myDGEList.filtered.norm, design, plot = FALSE)
fit <- lmFit(v.DEGList.filtered.norm, design)
#fit <- lmFit(v.DEGList.filtered.norm, design, block=targets$sample, correlation=corfit$consensus)
contrast.matrix <- makeContrasts(DVG = positive - negative,
                                 levels=design)


fits <- contrasts.fit(fit, contrast.matrix)
ebFit <- eBayes(fits)
myTopHits <- topTable(ebFit, adjust ="BH", coef=1, number=40000, sort.by="logFC")
```

```
myTopHits <- as_tibble(myTopHits, rownames = "geneSymbol")

ggplot(myTopHits, aes(y=-log10(adj.P.Val), x=logFC, text = paste("Symbol:", geneSymbol))) +
           geom_point(size=2) +
           ylim(-0.5,12) +
           geom_hline(yintercept = -log10(0.01), linetype="longdash", colour="grey", size=1) +
           geom_vline(xintercept = 1, linetype="longdash", colour="#BE684D", size=1) +
           geom_vline(xintercept = -1, linetype="longdash", colour="#2C467A", size=1) +
           labs(title="Volcano plot",
                subtitle = "DEGs for cbDVG+ vs cbDVG-",
                caption=paste0("produced on ", Sys.time()))
```




```
results <- decideTests(ebFit, method="global", adjust.method="BH", p.value=0.01, lfc=1)
summary(results)
```

```
##          DVG
## Down       6
## NotSig 20676
## Up       145
```

```
colnames(v.DEGList.filtered.norm$E) <- sampleLabels
diffGenes <- v.DEGList.filtered.norm$E[results[,1] !=0,]
```

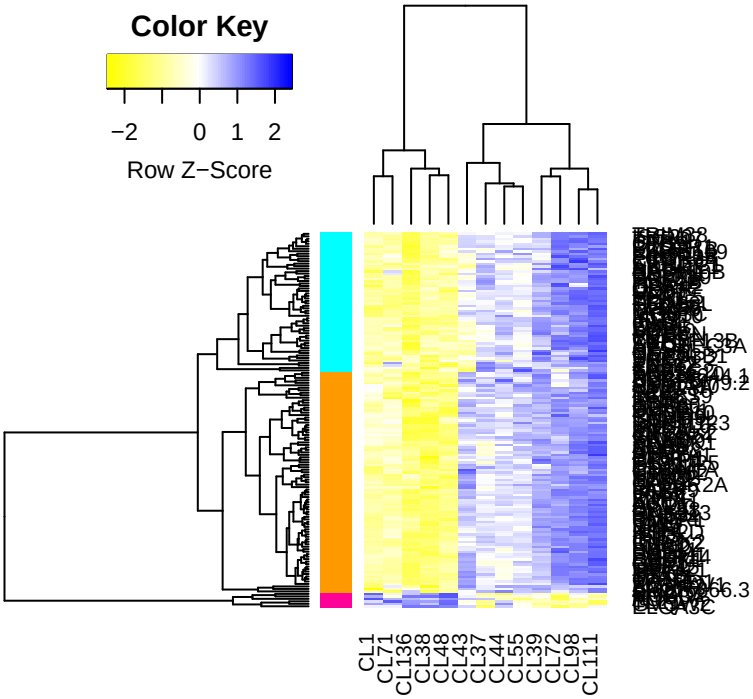
Step 7: modules

```
library(gplots) #the heatmap2 function in this package is a primary tool for making heatmaps
```

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

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

```
library(RColorBrewer) #need colors to make heatmaps
library(limma) #we only use limma in this script for the 'avearrays' function
myheatcolors2 <- colorRampPalette(colors=c("yellow","white","blue"))(100)
clustRows <- hclust(as.dist(1-cor(t(diffGenes), method="pearson")), method="complete") #cluster rows by
clustColumns <- hclust(as.dist(1-cor(diffGenes, method="spearman")), method="complete")
module.assign <- cutree(clustRows, k=3)
module.color <- rainbow(length(unique(module.assign)), start=0.1, end=0.9)
module.color <- module.color[as.vector(module.assign)]
heatmap.2(diffGenes,
          Rowv=as.dendrogram(clustRows),
          RowSideColors=module.color,
          col=myheatcolors2, scale='row',
          density.info="none", trace="none",
          cexRow=1, cexCol=1, margins=c(10,25))
```



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.