mRNAseq analysis: Yang 2019

Eleanor Williams
CSCI

01/03/21

Creating count matrix and normalising

```
meta = data.frame(type=c('0h','0h','12h','12h'),
    rep=c(1,2,1,2),
    id=c('SRR7624365','SRR7624366','SRR7624371','SRR7624372')
    )
meta$type = as.factor(meta$type)
listofsamples = paste(meta$id,'_counts.txt',sep='')
print(listofsamples)
```

```
## [1] "SRR7624365_counts.txt" "SRR7624366_counts.txt" "SRR7624371_counts.txt"
## [4] "SRR7624372_counts.txt"
```

```
print(head(read.csv(listofsamples[1],sep='\t',skip=1)))
```

```
##               Geneid           Chr
## 1 ENSMUSG00000102693             1
## 2 ENSMUSG00000064842             1
## 3 ENSMUSG00000051951 1;1;1;1;1;1;1
## 4 ENSMUSG00000102851             1
## 5 ENSMUSG00000103377             1
## 6 ENSMUSG00000104017             1
##                                                  Start
## 1                                              3073253
## 2                                              3102016
## 3 3205901;3206523;3213439;3213609;3214482;3421702;3670552
## 4                                              3252757
## 5                                              3365731
## 6                                              3375556
##                                                    End        Strand Length
## 1                                              3074322             +   1070
## 2                                              3102125             +    110
## 3 3207317;3207317;3215632;3216344;3216968;3421901;3671498 -;-;-;-;-;-;-   6094
## 4                                              3253236             +    480
## 5                                              3368549             -   2819
## 6                                              3377788             -   2233
##   SRR7624365__Aligned.out.bam
```

```
## 1                           1
## 2                           0
## 3                           3
## 4                           0
## 5                           0
## 6                           0
```

```
head(read.csv(listofsamples[1],sep='\t',skip=1)[,c(1,7)])
```

```
##               Geneid SRR7624365__Aligned.out.bam
## 1 ENSMUSG00000102693                           1
## 2 ENSMUSG00000064842                           0
## 3 ENSMUSG00000051951                           3
## 4 ENSMUSG00000102851                           0
## 5 ENSMUSG00000103377                           0
## 6 ENSMUSG00000104017                           0
```

```
cts = data.frame(gene_id = read.csv(listofsamples[1],sep='\t',skip=1)[,1])
rownames(cts) = cts$gene_id

for (i in meta$id){
  print(head(i,10))
  currentfile = read.csv(paste(i,'_counts.txt',sep=''),sep='\t',skip=1)
  columns = colnames(currentfile)
  cts[,i]=currentfile[,7]
}
```

```
## [1] "SRR7624365"
## [1] "SRR7624366"
## [1] "SRR7624371"
## [1] "SRR7624372"
```

```
cts = subset(cts,select=-c(gene_id))

cts = cts[rowSums(cts) > 0,]
```

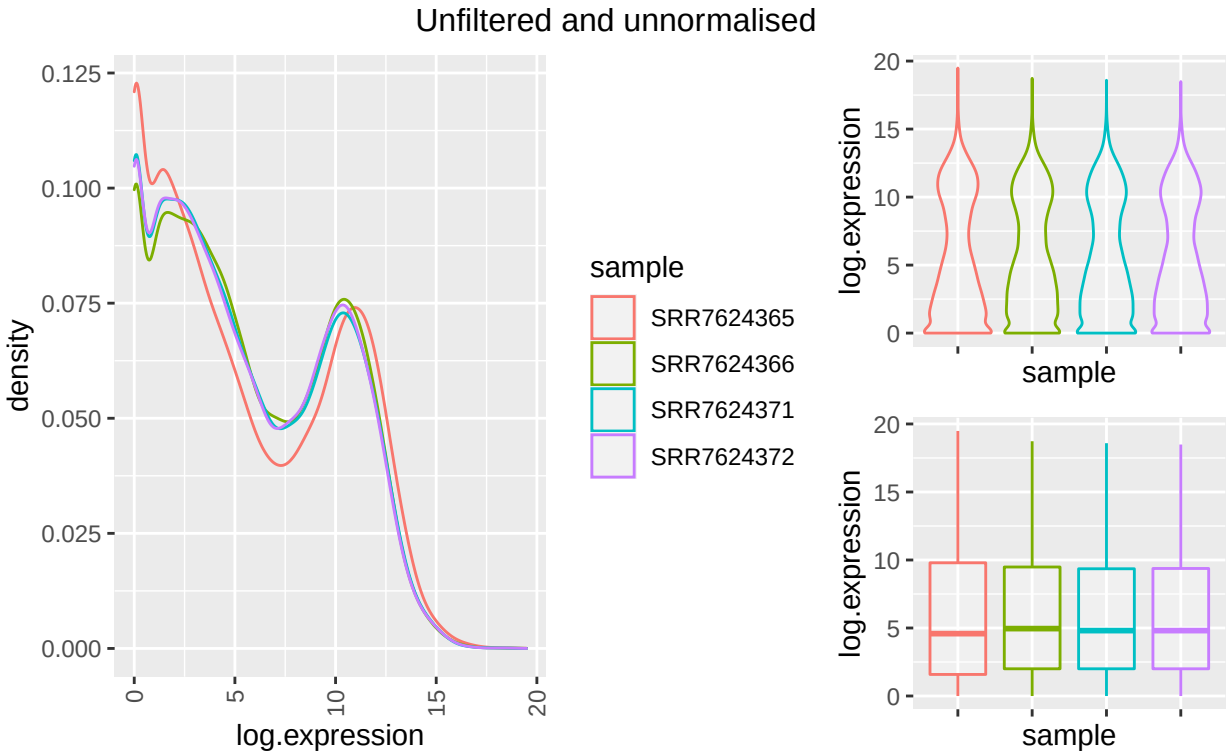
```
expression.threshold <- 20
cts.filtered = cts
cts.filtered[cts.filtered<expression.threshold]<-expression.threshold
cts.filtered = cts.filtered[rowSums(cts.filtered)>expression.threshold*ncol(cts.filtered),]
```

```
cts.qnorm.unfiltered=data.frame(normalize.quantiles(as.matrix(cts)),row.names=rownames(cts))
colnames(cts.qnorm.unfiltered)=colnames(cts)

cts.qnorm.filtered=data.frame(normalize.quantiles(as.matrix(cts.filtered)),row.names=rownames(cts.filte
colnames(cts.qnorm.filtered)=colnames(cts.filtered)
```

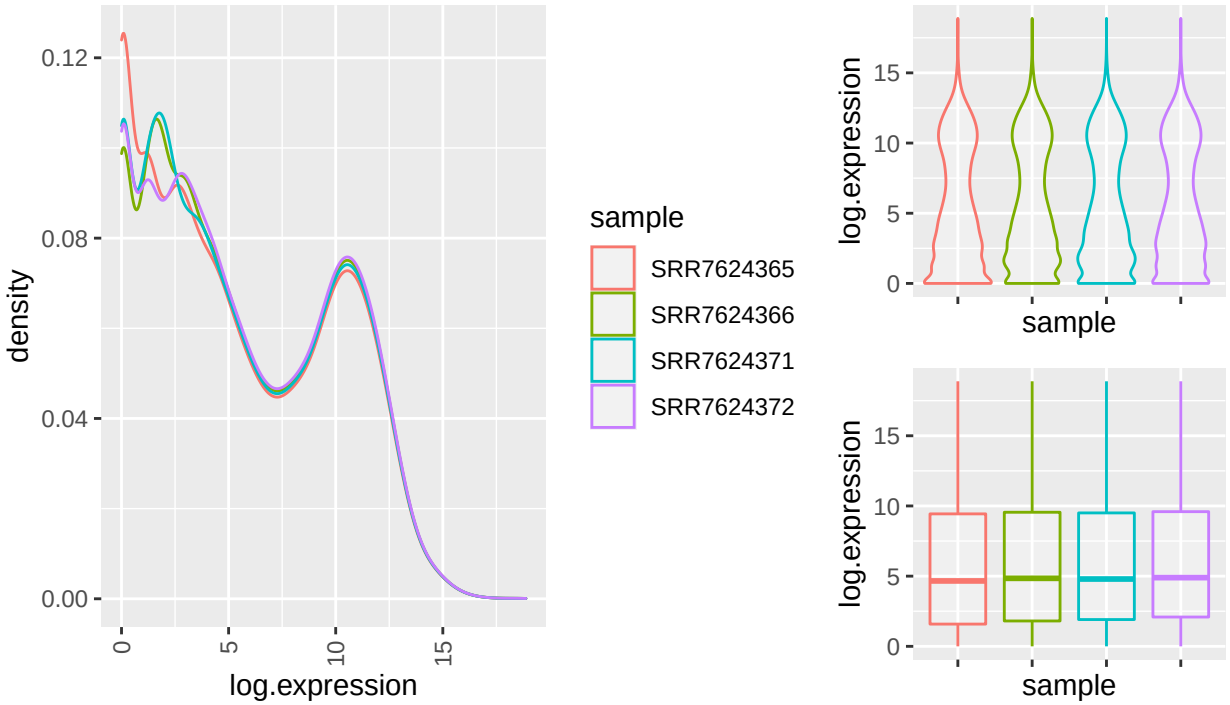
Quality Control

Visualising distribution of abundance per sample

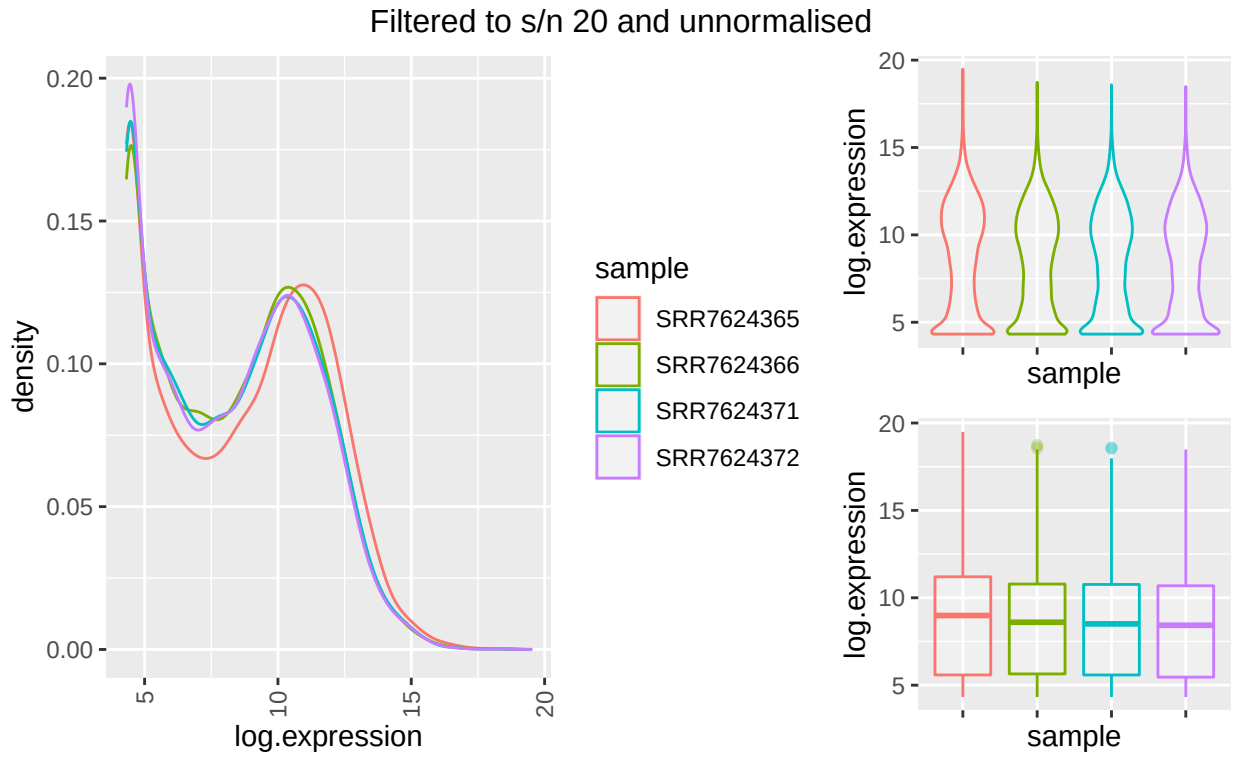
```
qc.plots<-function(cts,title){
cts.tidy = pivot_longer(cts, cols=colnames(cts), names_to='sample', values_to='expression')
# we now remove 0 counts to create more understandable visualizations
cts.tidy$expression[cts.tidy$expression == 0] = NA
cts.tidy$log.expression=log2(cts.tidy$expression)

a<-ggplot(drop_na(cts.tidy), aes(x=log.expression, color=sample)) +
  geom_density(alpha=0.3)+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  theme(plot.title = element_text(size=10))
b<-ggplot(drop_na(cts.tidy), aes(x=sample, y=log.expression,color=sample)) +
  geom_violin(alpha=0.3) +
  theme(legend.position = "none")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  theme(plot.title = element_text(size=10))+
  theme(axis.text.x=element_blank())
c<-ggplot(drop_na(cts.tidy), aes(x=sample, y=log.expression,color=sample)) +
  geom_boxplot(alpha=0.3) +
  theme(legend.position = "none")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  theme(plot.title = element_text(size=10))+
  theme(axis.text.x=element_blank())
  grid.arrange(a,arrangeGrob(b,c),nrow=1,top=title,widths=2:1)
}
qc.plots(cts,'Unfiltered and unnormalised')
```



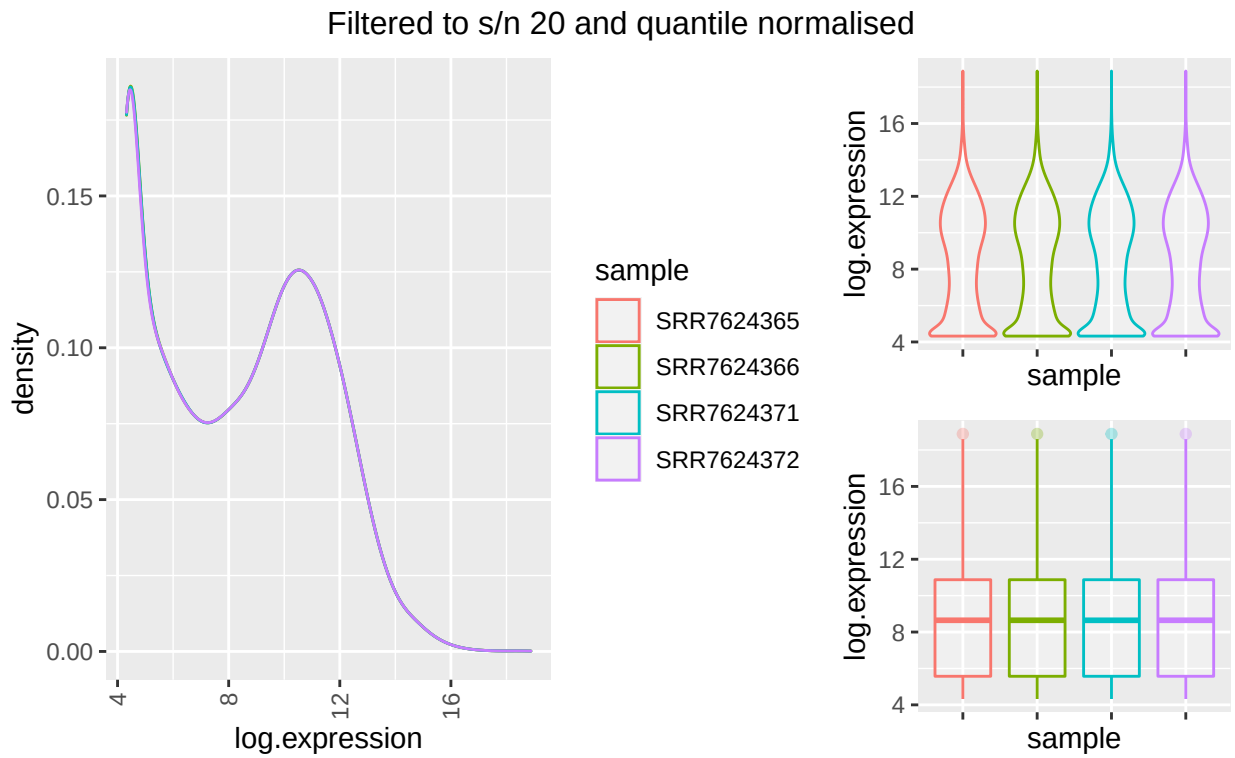

```
qc.plots(cts.qnorm.unfiltered,'Unfiltered and quantile normalised')
```



```
qc.plots(cts.filtered,'Filtered to s/n 20 and unnormalised')
```

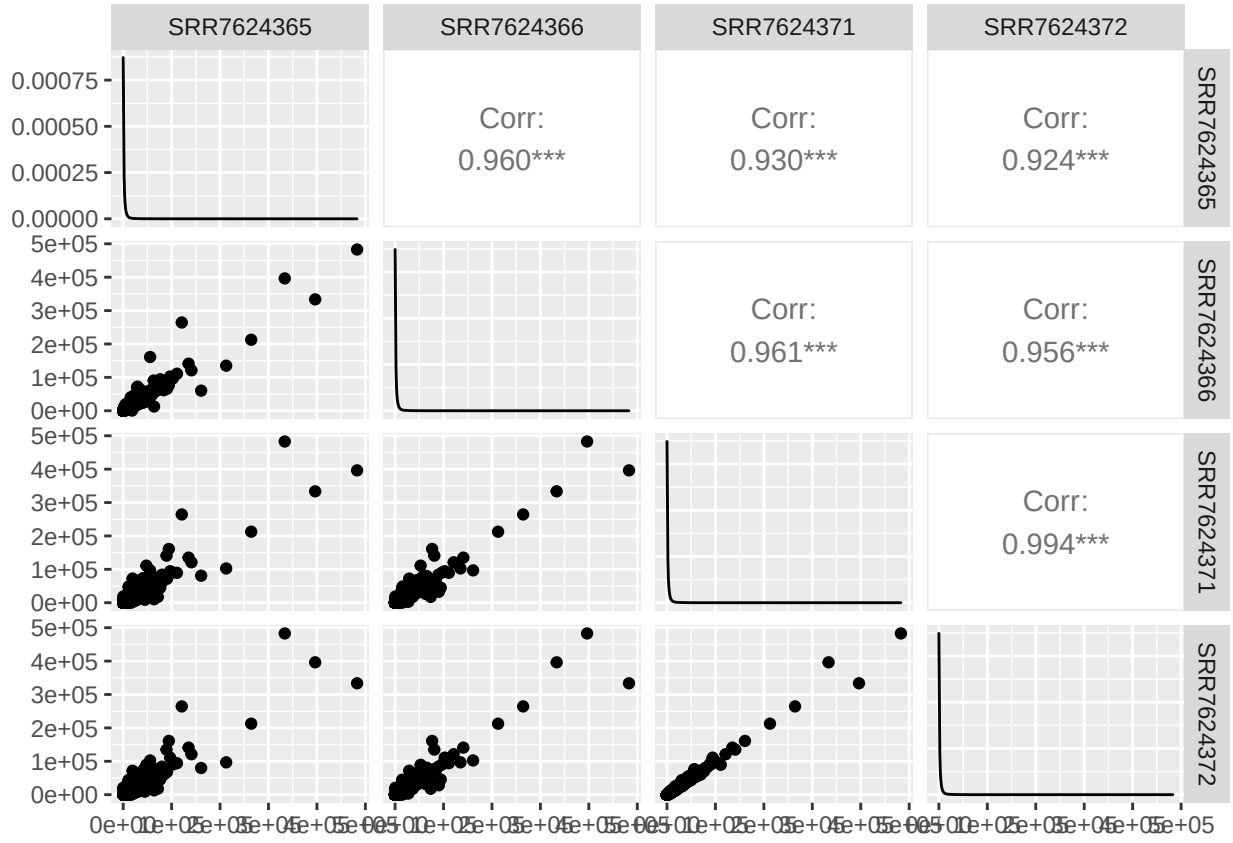



```
qc.plots(cts.qnorm.filtered,'Filtered to s/n 20 and quantile normalised')
```



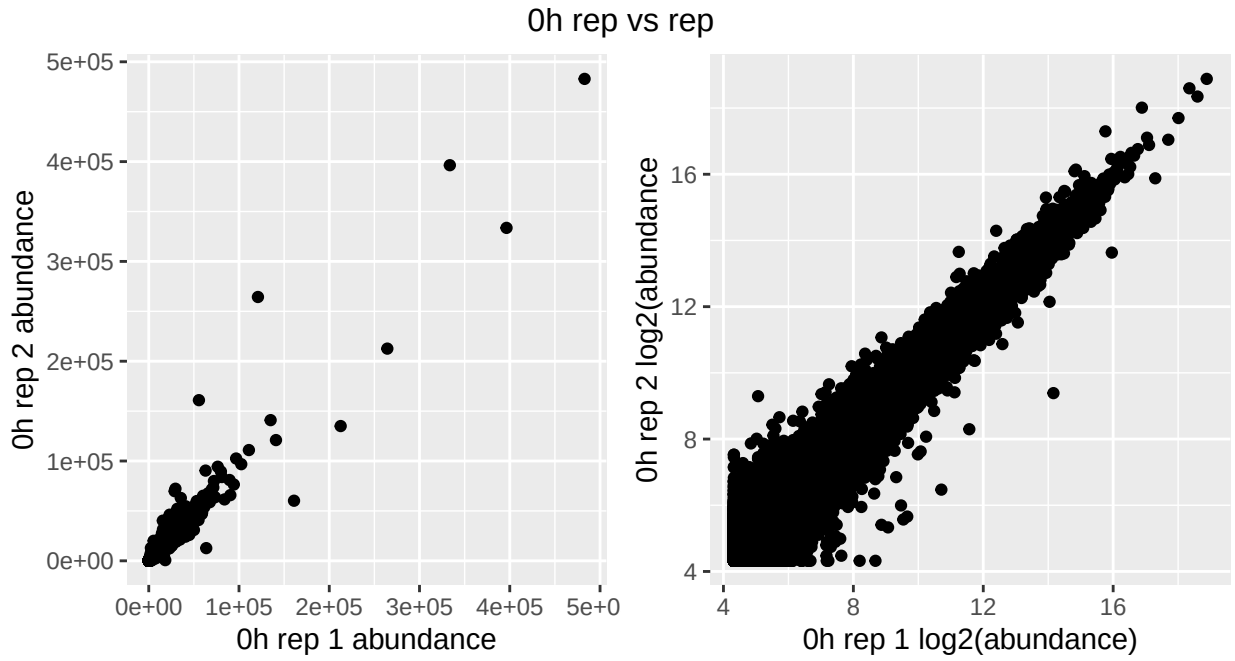
ggpairs

```
ggpairs(cts.qnorm.filtered)
```

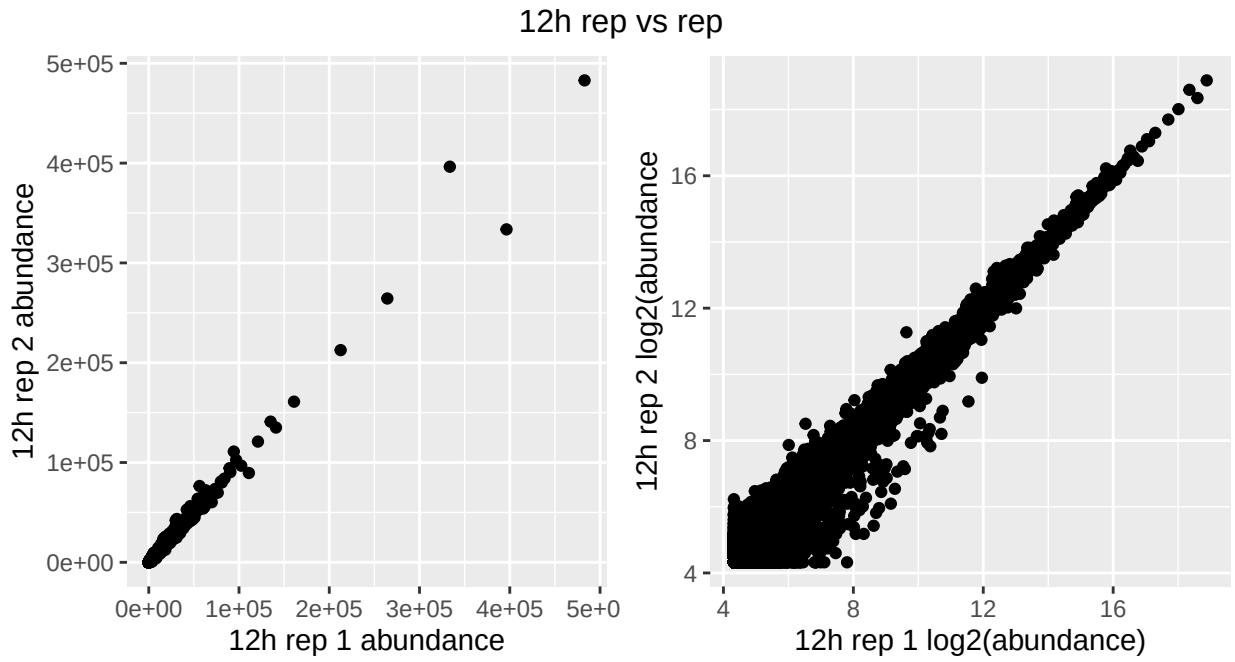


Comparing replicates

```
grid.arrange(ggplot(cts.qnorm.filtered, aes(x=SRR7624365, y=SRR7624366)) +
        geom_point() +
        xlab('0h rep 1 abundance') +
        ylab('0h rep 2 abundance'),
        ggplot(cts.qnorm.filtered, aes(x=log2(SRR7624365), y=log2(SRR7624366))) +
        geom_point() +
        xlab('0h rep 1 log2(abundance)') +
        ylab('0h rep 2 log2(abundance'),nrow=1,top='0h rep vs rep')
```




```
grid.arrange(ggplot(cts.qnorm.filtered, aes(x=SRR7624371, y=SRR7624372)) +
        geom_point() +
        xlab('12h rep 1 abundance') +
        ylab('12h rep 2 abundance'),
        ggplot(cts.qnorm.filtered, aes(x=log2(SRR7624371), y=log2(SRR7624372))) +
        geom_point() +
        xlab('12h rep 1 log2(abundance)') +
        ylab('12h rep 2 log2(abundance'),nrow=1,top='12h rep vs rep')
```



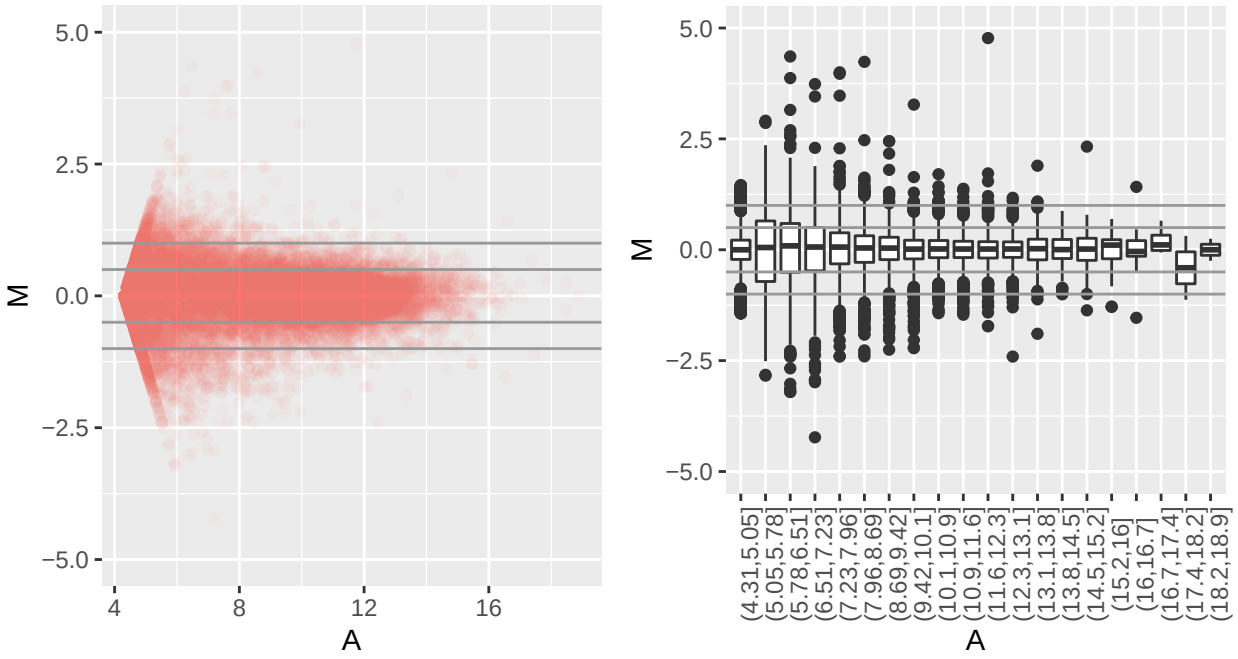
MA plots

```
ma.plot = function(counts, meta, i, j, lower.lim=NA, upper.lim=NA, log.transformed=FALSE){
  main.title = paste0(meta$id[i], ' v ', meta$id[j])
  sub.title = paste(meta$type[i], meta$rep[i], 'v',
                    meta$type[j], meta$rep[j])
  # if already logtransformed we don't need to do it again
  if (log.transformed == TRUE){
    l1 = counts[,i]
    l2 = counts[,j]
  } else {
    # mask away the zeros
    zero.mask = !(counts[,i] == 0 | counts[,j] == 0)
    l1 = log2(counts[zero.mask, i])
    l2 = log2(counts[zero.mask, j])
  }
  m = l1 - l2
  a = 0.5 * (l1 + l2)
  data = data.frame(A = a, M = m)
  p = ggplot(data=data, aes(x=A, y=M, color='red', fill='red')) +
    geom_point(alpha=0.05)+
    theme(legend.position = "none") +
    geom_hline(yintercept=0.5,colour='gray60')+
    geom_hline(yintercept=-0.5,colour='gray60')+
    geom_hline(yintercept=1,colour='gray60')+
    geom_hline(yintercept=-1,colour='gray60')
  a.binned = cut(a, 20)
  data.binned = data.frame(A = a.binned, M = m)
  q = ggplot(data=data.binned) +
    geom_boxplot(aes(A, M)) +
    theme(axis.text.x=element_text(angle=90))+
    theme(legend.position = "none")  +
    geom_hline(yintercept=0.5,colour='gray60')+
    geom_hline(yintercept=-0.5,colour='gray60')+
    geom_hline(yintercept=1,colour='gray60')+
    geom_hline(yintercept=-1,colour='gray60')

  # add ylim only if one of upper.lim, lower.lim is non-NA
  if (!is.na(lower.lim) | !is.na(upper.lim)){
    p = p + ylim(lower.lim, upper.lim)
    q = q + ylim(lower.lim, upper.lim)
  }
  grid.arrange(p, q, ncol = 2, top=paste0(main.title, '\n', sub.title))
}
```

```
llim = -5
ulim = 5
for (i in 1:2){
  ma.plot(cts.qnorm.filtered, meta, (2*i)-1, 2*i, llim, ulim)
  }
```

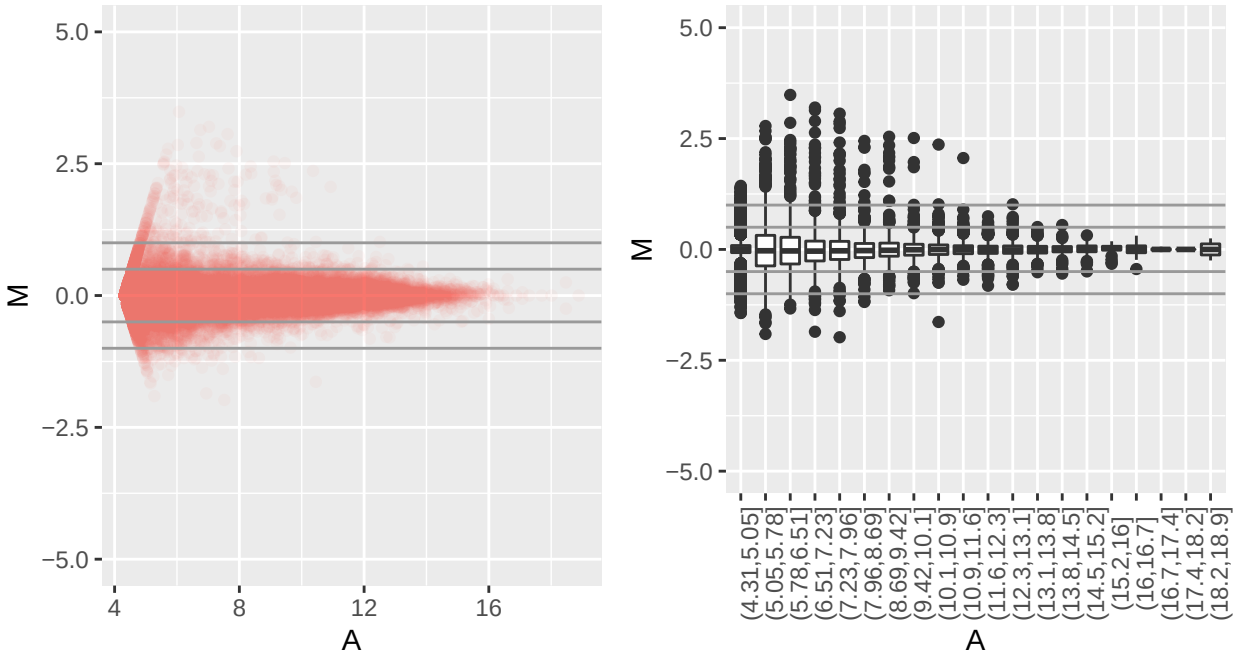
SRR7624365 v SRR7624366

0h 1 v 0h 2



SRR7624371 v SRR7624372

12h 1 v 12h 2



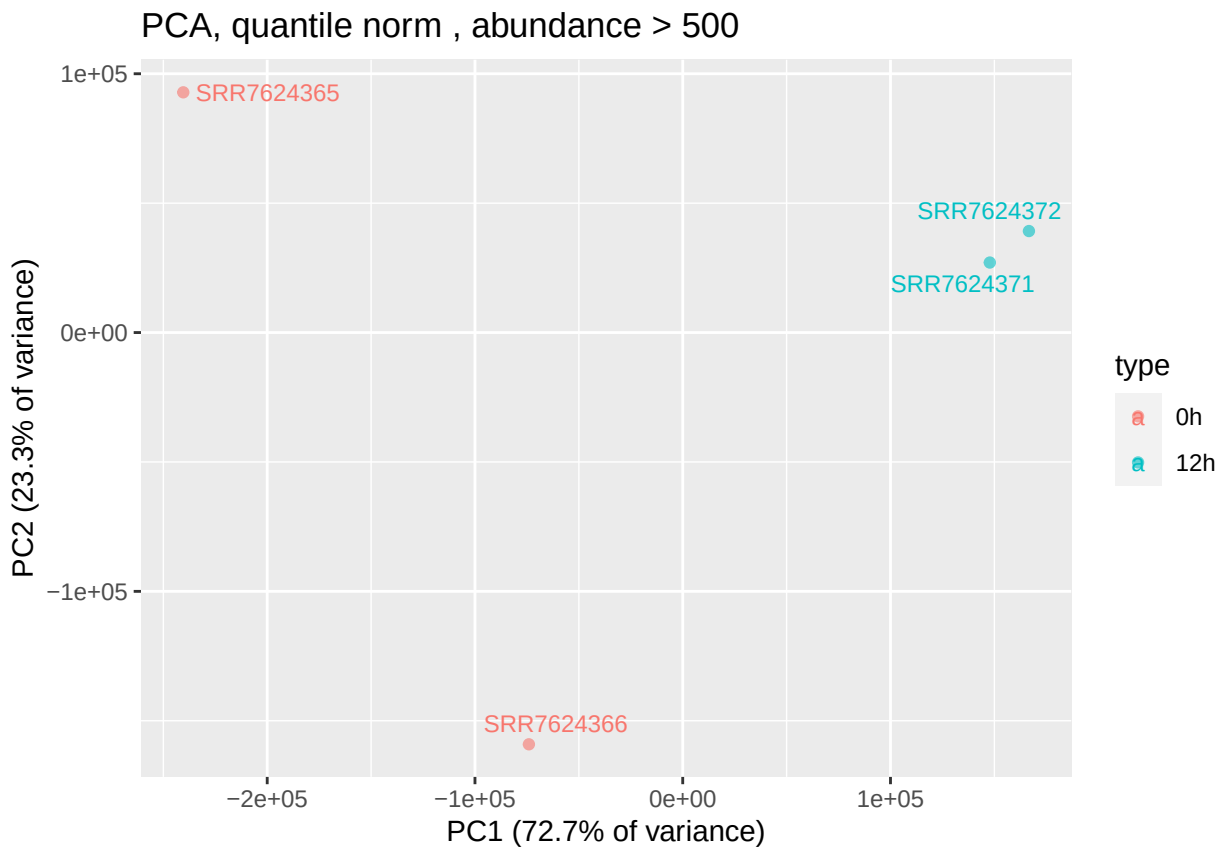
PCA

```
plot.pca = function(pca.results, meta.table, title){
  data = data.frame(PC1=pca.results$x[,1], PC2=pca.results$x[,2])
  data = cbind(data, meta.table)
  eigs = pca.results$sdev ** 2
  eigs = eigs / sum(eigs)
  xlab = paste0('PC1 (', format(eigs[1]*100, digits=3), '% of variance)')
  ylab = paste0('PC2 (', format(eigs[2]*100, digits=3), '% of variance)')
  ggplot(data=data, aes(x=PC1, y=PC2, color=type)) +
    geom_text_repel(data=data, aes(x=PC1, y=PC2, label=id), size=3) +
    geom_point(alpha=0.6) +
    labs(title=title, x=xlab, y=ylab)
}
```

```
pdf('incrementalPCA.pdf')
for (i in c(50,seq(100,2000,by=100))){
  expression.threshold = i
  keep.features = apply(cts, 1, min)>expression.threshold
  cts.filtered = cts[keep.features,]
  cts.qnorm=data.frame(normalize.quantiles(as.matrix(cts.filtered)),row.names=rownames(cts.filtered))
  colnames(cts.qnorm)=colnames(cts.filtered)
  pca.norm = prcomp(t(cts.qnorm))
  print(plot.pca(pca.norm, meta, paste('PCA, quantile norm , abundance > ',i,sep='')))
}
dev.off()
```

```
## pdf
##   2
```

```
i=500
expression.threshold = i
keep.features = apply(cts, 1, min)>expression.threshold
cts.filtered = cts[keep.features,]
cts.qnorm=data.frame(normalize.quantiles(as.matrix(cts.filtered)),row.names=rownames(cts.filtered))
colnames(cts.qnorm)=colnames(cts.filtered)
pca.norm = prcomp(t(cts.qnorm))
print(plot.pca(pca.norm, meta, paste('PCA, quantile norm , abundance > ',i,sep='')))
```



JSI

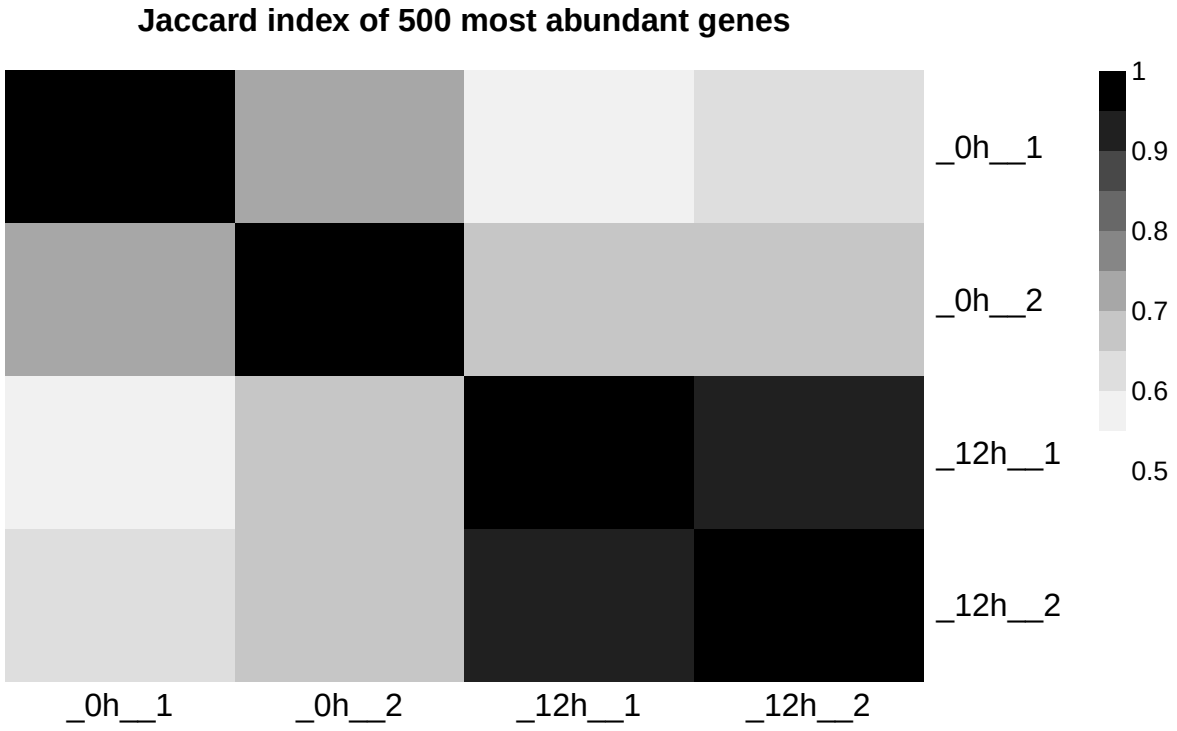
```
jaccard.index = function(a, b){
  if ((length(a) == 0) & (length(b) == 0)){
    return(1)
  } else{
    u = length(union(a,b))
    i = length(intersect(a,b))
    return(i/u)
  }
}

leaf.labels=paste(meta$barcode,meta$type,meta$patient,meta$rep,sep='_')
jaccard.heatmap = function(counts, n.abundant, labels){
  #  colnames_counts <- c(t(inner(meta$type, meta$n_replicate, paste, sep = "_")))
  colnames_counts=paste0(meta$patient,meta$type,meta$rep)
  labels=labels[order(colnames_counts)]
  #  colnames(reorderedcounts)=colnames_counts
  counts=counts[,order(colnames_counts)]
  n.samples = ncol(counts)
  hm = matrix(nrow=n.samples, ncol=n.samples)
  hm[] = 0
  for (i in 1:n.samples){
    for (j in 1:i){
```

```
      i.gene.indices = order(counts[,i], decreasing=TRUE)[1:n.abundant]
      j.gene.indices = order(counts[,j], decreasing=TRUE)[1:n.abundant]
      hm[i, j] = jaccard.index(i.gene.indices, j.gene.indices)
      hm[j, i] = hm[i, j]
    }
  }
  title = paste0('Jaccard index of ', n.abundant, ' most abundant genes')
  aheatmap(hm,
           color='Greys',
           Rowv = NA,
           Colv = NA,
           labRow=labels,
           labCol=labels,
           main=title,
           breaks=c(0.5,0.55,0.6,0.65,0.7,0.75,0.8,0.85,0.9,0.95,1),
           treeheight=0)
}
```

```
pdf('Jaccardplots.pdf')
n.abundances = c(2000, 1000, 500, 200, 100, 50)
for (n in n.abundances){
  jaccard.heatmap(cts.qnorm.filtered, n, leaf.labels)
}
dev.off()
```

```
## pdf
##   2
```

```
jaccard.heatmap(cts.qnorm.filtered, 500, leaf.labels)
```



Differential Expression

```
expression.threshold <- 20
cts.filtered = cts
cts.filtered[cts.filtered<expression.threshold]<-expression.threshold
cts.filtered = cts.filtered[rowSums(cts.filtered)>expression.threshold*ncol(cts.filtered),]

cts.qnorm.unfiltered=data.frame(normalize.quantiles(as.matrix(cts)),
                                row.names=rownames(cts))
colnames(cts.qnorm.unfiltered)=colnames(cts)

cts.qnorm.filtered=data.frame(normalize.quantiles(as.matrix(cts.filtered)),
                              row.names=rownames(cts.filtered))
colnames(cts.qnorm.filtered)=colnames(cts.filtered)
```

edgeR

```
edg = DGEList(cts.qnorm.filtered,group=c(1,1,2,2))
edg = estimateDisp(edg)
```

```
## Using classic mode.
```

```
exact =exactTest(edg,dispersion=0.2)
exact$table$adjustedp = p.adjust(exact$table$PValue,method='BH')
print(length(rownames(exact$table[abs(exact$table$logFC) > 0.5 & exact$table$adjustedp<0.05,])))
```

```
## [1] 510
```

```
edger_genes = rownames(exact$table[abs(exact$table$logFC) > 0.5 & exact$table$adjustedp<0.05,])
```

DESeq2

```
ds = DESeqDataSetFromMatrix(countData = round(cts.qnorm.filtered), colData = meta, design = ~ type)
```

```
## converting counts to integer mode
```

```
ds = ds[rowSums(counts(ds)) > 0,]
ds$sizeFactor=rep(1:1,ncol(cts.qnorm.filtered))
ds = DESeq(ds)
```

```
## using pre-existing size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
cts.ma = log2(counts(ds, normalized=TRUE) + 1)
cts.ds2.norm = as.data.frame(cts.ma)
ds.results = na.omit(results(ds, contrast=c('type','12h','0h'),
                             lfcThreshold=1.5,
                             altHypothesis='greaterAbs',
                             independentFiltering=TRUE ,
                             alpha=0.05))
deseq_genes = rownames(ds.results[abs(ds.results$log2FoldChange)>0.5 & ds.results$padj<0.05,])
print(length(deseq_genes))
```

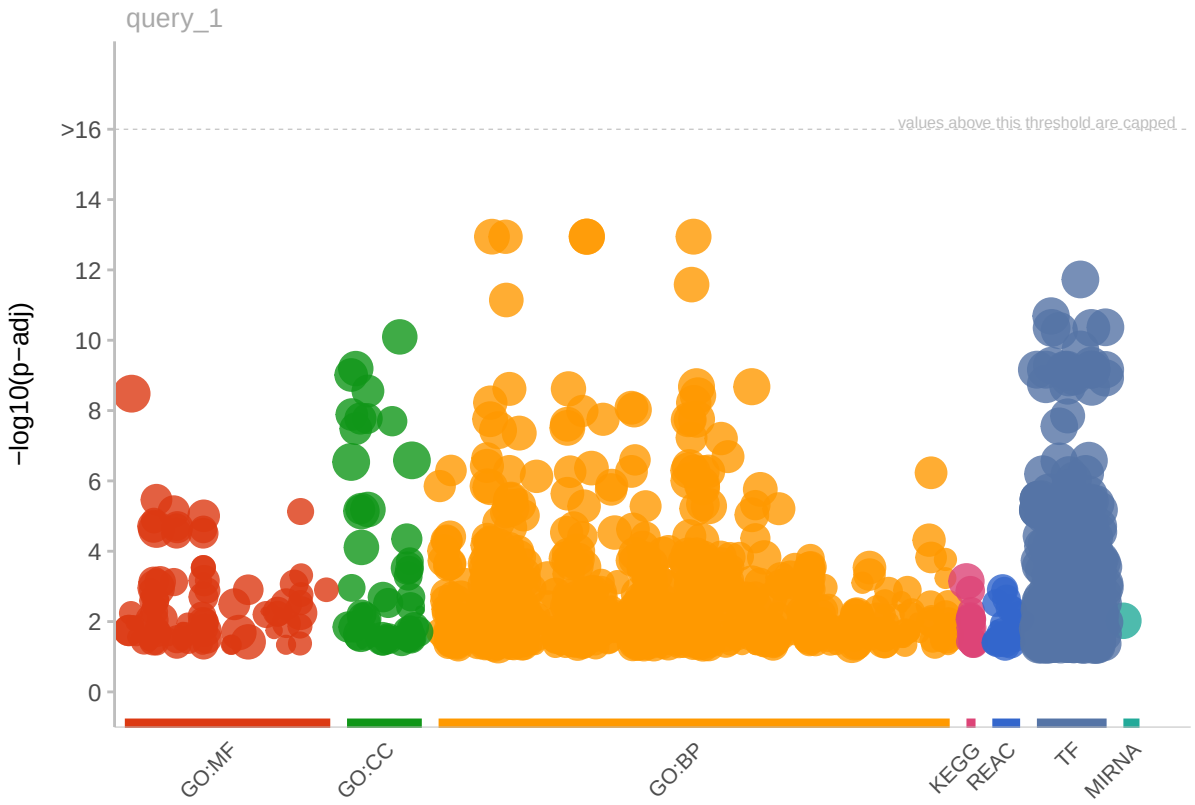
```
## [1] 295
```

Enrichment

```
gprofiler_results = gprofiler2::gost(intersect(edger_genes,deseq_genes),
                                     organism='mmusculus',
                                     custom_bg = rownames(cts.filtered),
                                     sources=c('GO:BP','GO:MF','GO:CC','KEGG','REAC','TF','MIRNA'),
                                     correction_method='fdr')
```

```
## Detected custom background input, domain scope is set to 'custom'
```

```
gostplot(gprofiler_results, capped = TRUE, interactive = FALSE)
```




```
print(head(gprofiler_results$result))
```

```
##     query significant      p_value term_size query_size intersection_size
## 1 query_1        TRUE 1.133794e-13      4679        291              141
## 2 query_1        TRUE 1.133794e-13      4315        291              134
## 3 query_1        TRUE 1.133794e-13      5000        291              148
## 4 query_1        TRUE 1.133794e-13      3969        291              127
## 5 query_1        TRUE 1.139944e-13      1749        291               77
## 6 query_1        TRUE 2.617911e-12      3551        291              115
##   precision     recall    term_id source                        term_name
## 1 0.4845361 0.03013464 GO:0032502  GO:BP              developmental process
## 2 0.4604811 0.03105446 GO:0048856  GO:BP   anatomical structure development
## 3 0.5085911 0.02960000 GO:0032501  GO:BP   multicellular organismal process
```

```
## 4 0.4364261 0.03199798 GO:0007275  GO:BP multicellular organism development
## 5 0.2646048 0.04402516 GO:0009605  GO:BP      response to external stimulus
## 6 0.3951890 0.03238524 GO:0048731  GO:BP                system development
##   effective_domain_size source_order               parents
## 1                 18036         8845             GO:0008150
## 2                 18036        15211             GO:0032502
## 3                 18036         8844             GO:0008150
## 4                 18036         3186 GO:0032501, GO:0048856
## 5                 18036         3997             GO:0050896
## 6                 18036        15094 GO:0007275, GO:0048856
```