sRNAseq analysis: Wang 2019

Eleanor Williams
CSCI

01/03/21

Creating count matrix and normalising

```
meta = data.frame(cell=c(1,2,3,4),
    id=c("SRR7124085","SRR7124086","SRR7124087","SRR7124088")
    )
meta$cell = as.factor(meta$cell)
```

```
cts = read.csv('counts.csv')
rownames(cts)=cts$miRNA
cts = cts[3:6]
```

```
expression.threshold <- 10
cts.filtered = cts
cts.filtered[cts.filtered<expression.threshold]<-expression.threshold
cts.filtered = cts.filtered[rowSums(cts.filtered)>expression.threshold*ncol(cts.filtered),]
```

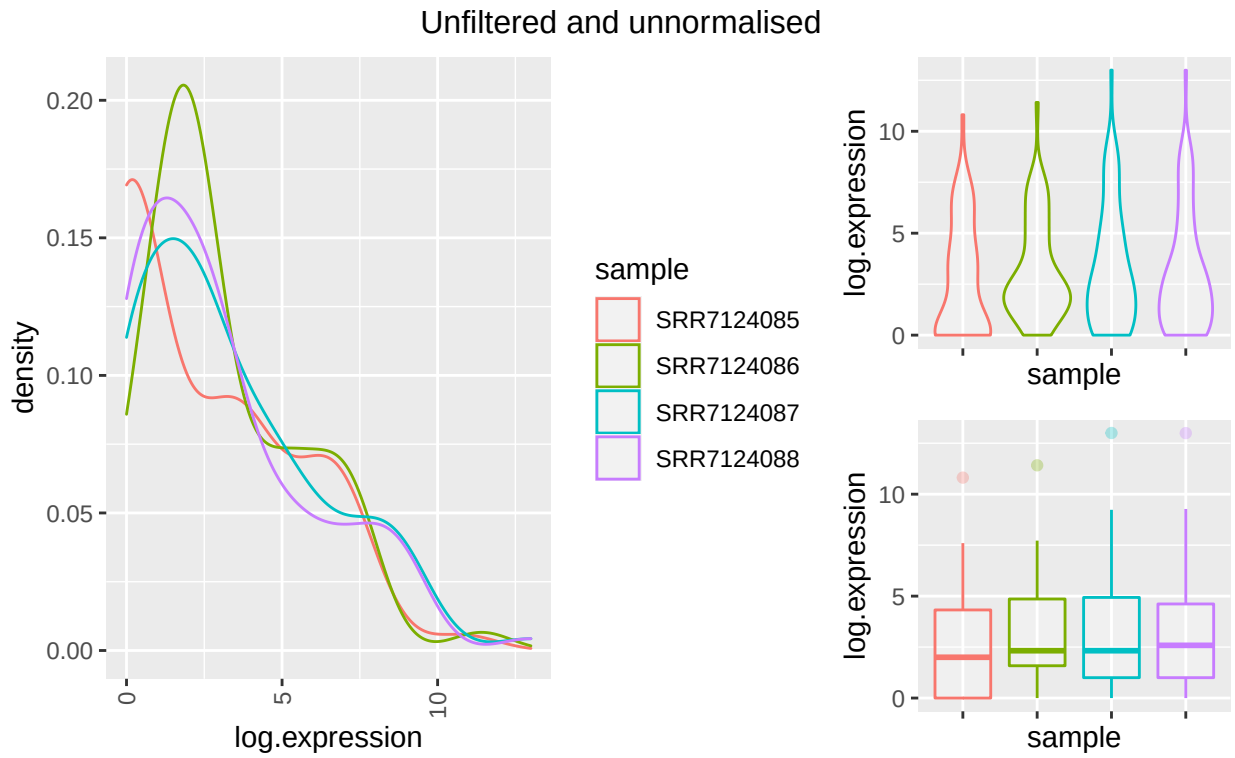
```
cts.qnorm.unfiltered=data.frame(normalize.quantiles(as.matrix(cts)),
                                row.names=rownames(cts))
colnames(cts.qnorm.unfiltered)=colnames(cts)

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

Quality Control

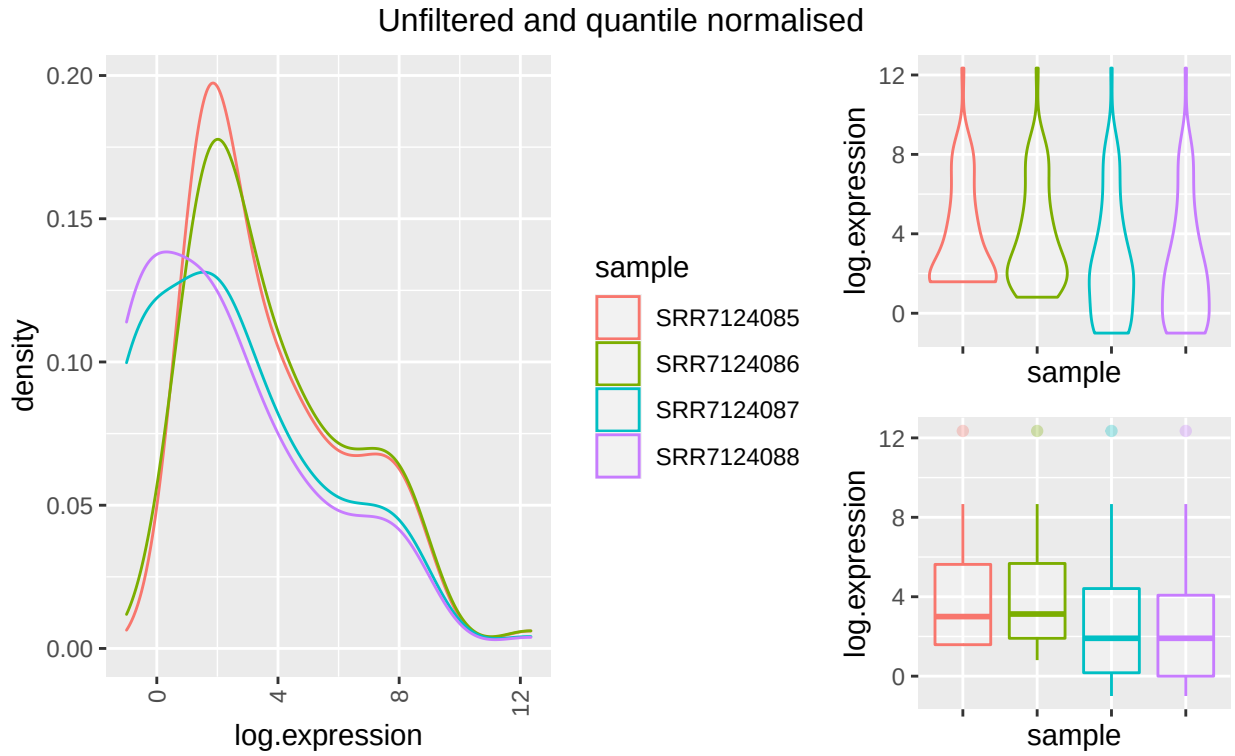
Visualising distribution of abundance per sample

```
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)
```

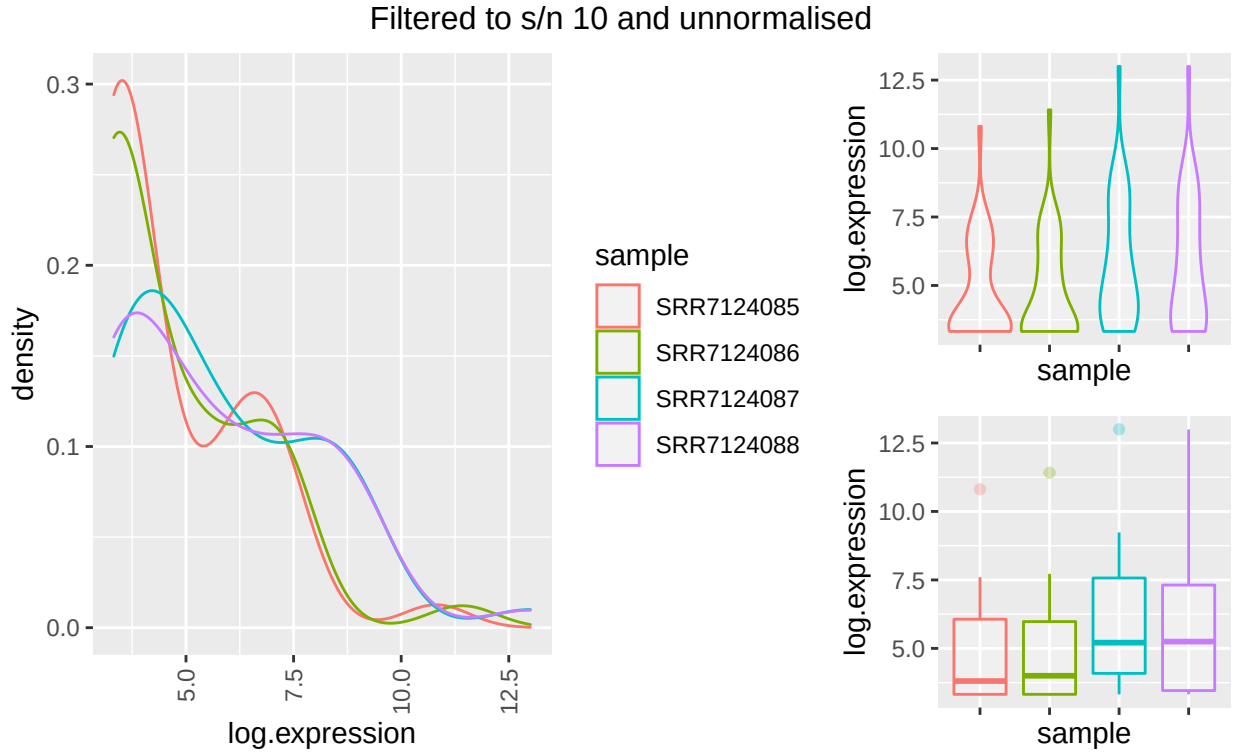
```
a<-ggplot(drop_na(cts.tidy), aes(x=log.expression, color=sample)) +
  geom_density(alpha=0.3)+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))+
  theme(plot.title = element_text(size=10))
b<-ggplot(drop_na(cts.tidy), aes(x=sample, y=log.expression,color=sample)) +
  geom_violin(alpha=0.3) +
  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')
```



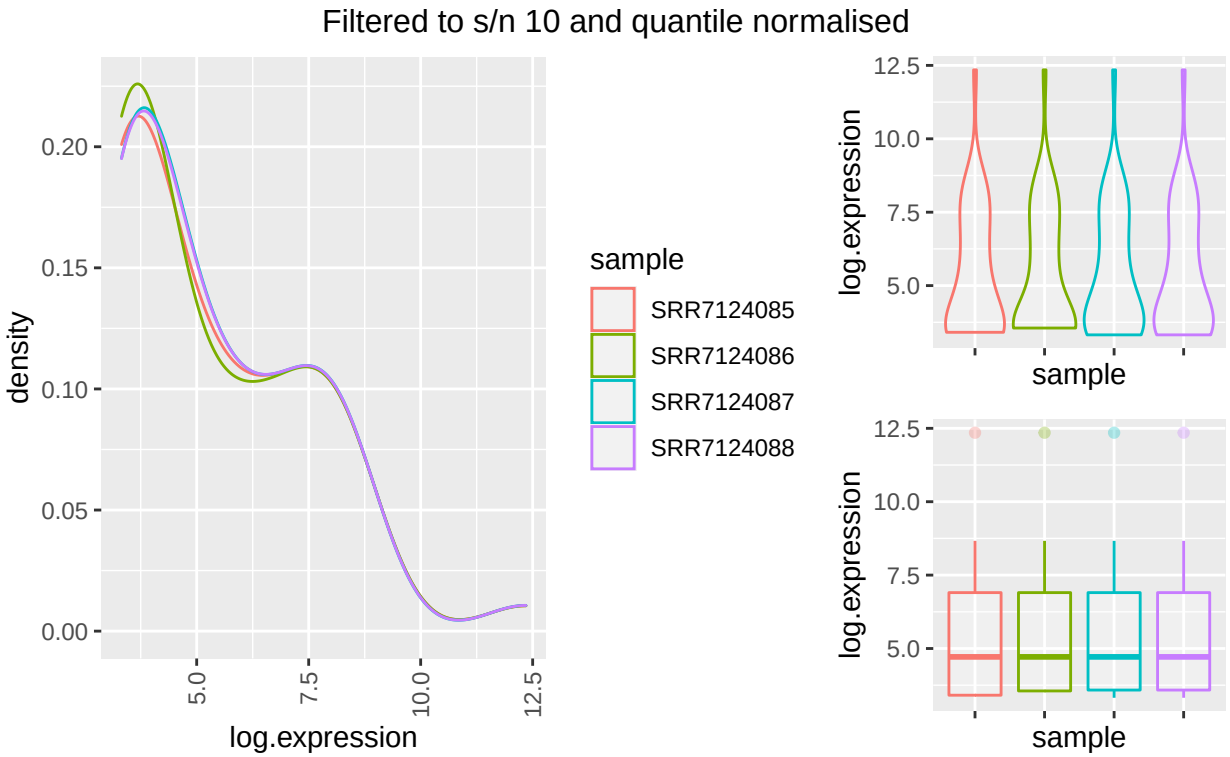

```
qc.plots(cts.qnorm.unfiltered,'Unfiltered and quantile normalised')
```



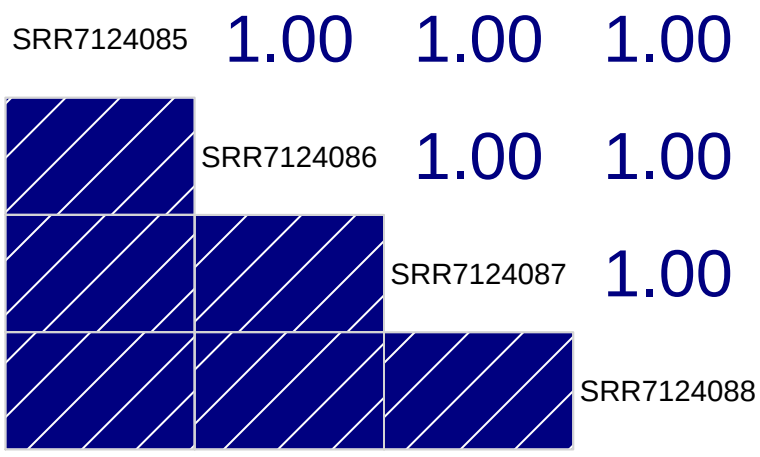

```
qc.plots(cts.filtered,'Filtered to s/n 10 and unnormalised')
```



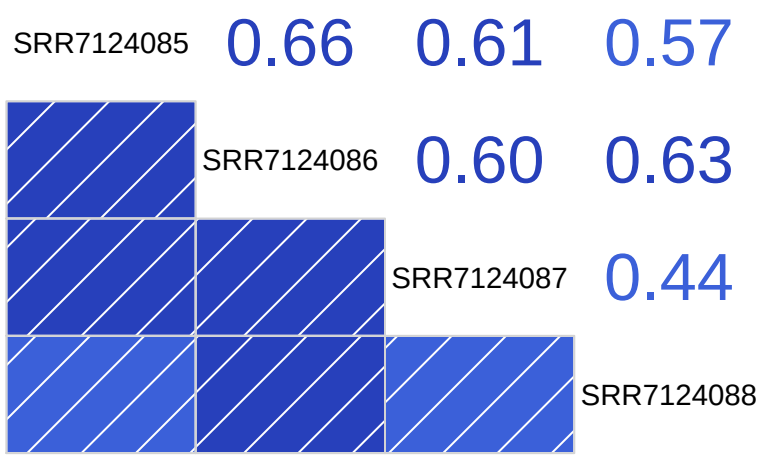
```
qc.plots(cts.qnorm.filtered,'Filtered to s/n 10 and quantile normalised')
```



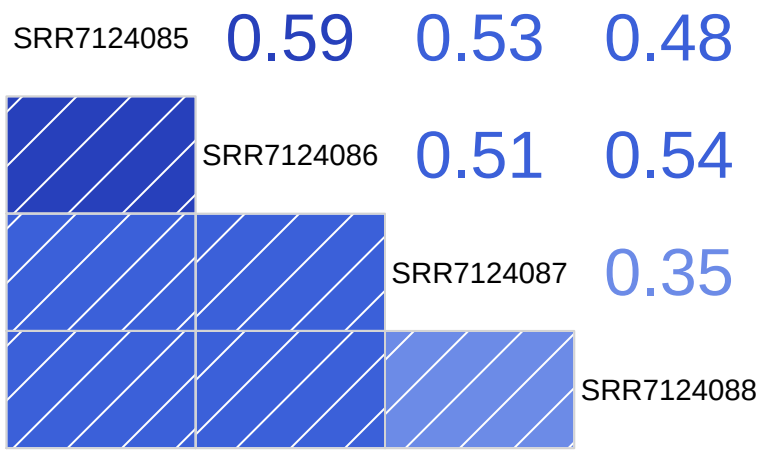

```
cor.matrix <- cor((cts),method = 'pearson')
corrgram::corrgram(cor.matrix, order=FALSE,
         upper.panel=panel.cor)
```



```
cor.matrix <- cor((cts),method = 'spearman')
corrgram::corrgram(cor.matrix, order=FALSE,
         upper.panel=panel.cor)
```

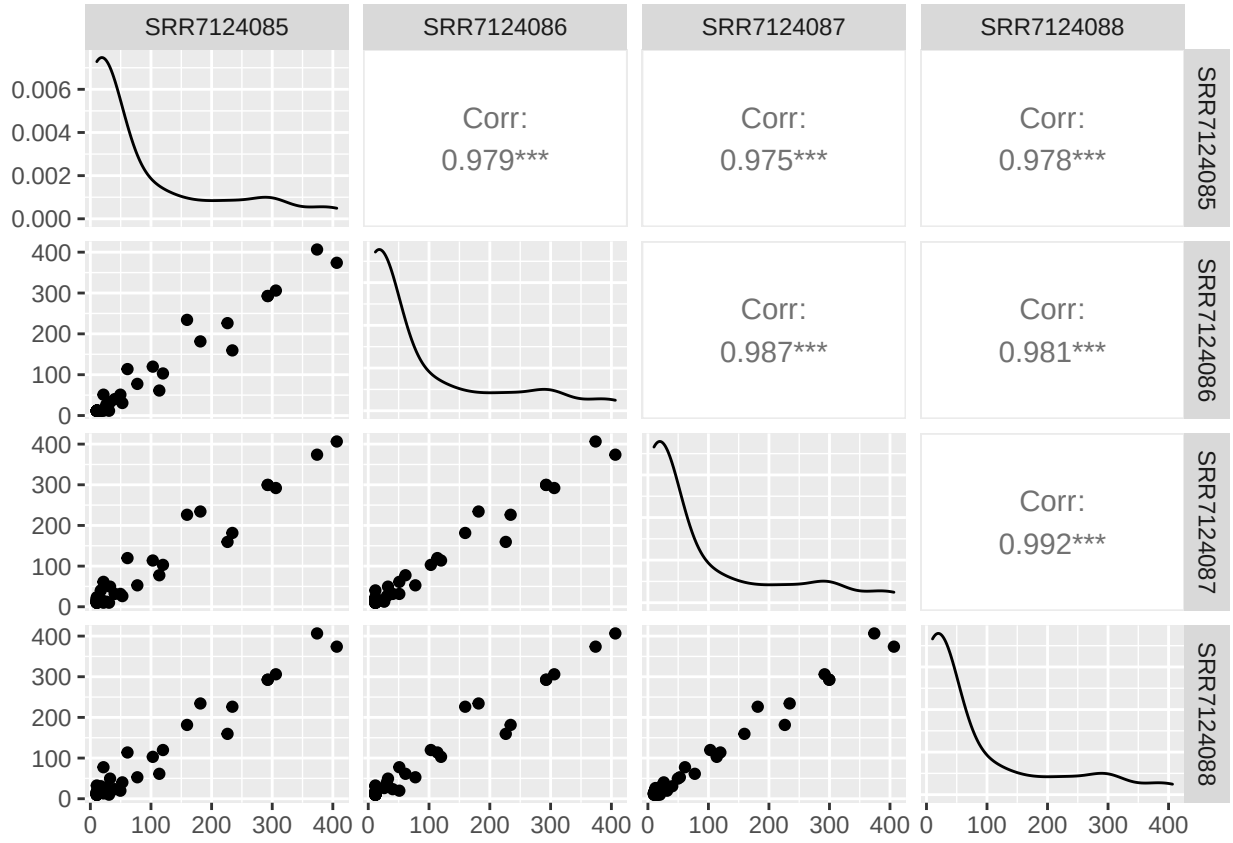



```
cor.matrix <- cor((cts),method = 'kendall')
corrgram::corrgram(cor.matrix, order=FALSE,
         upper.panel=panel.cor)
```

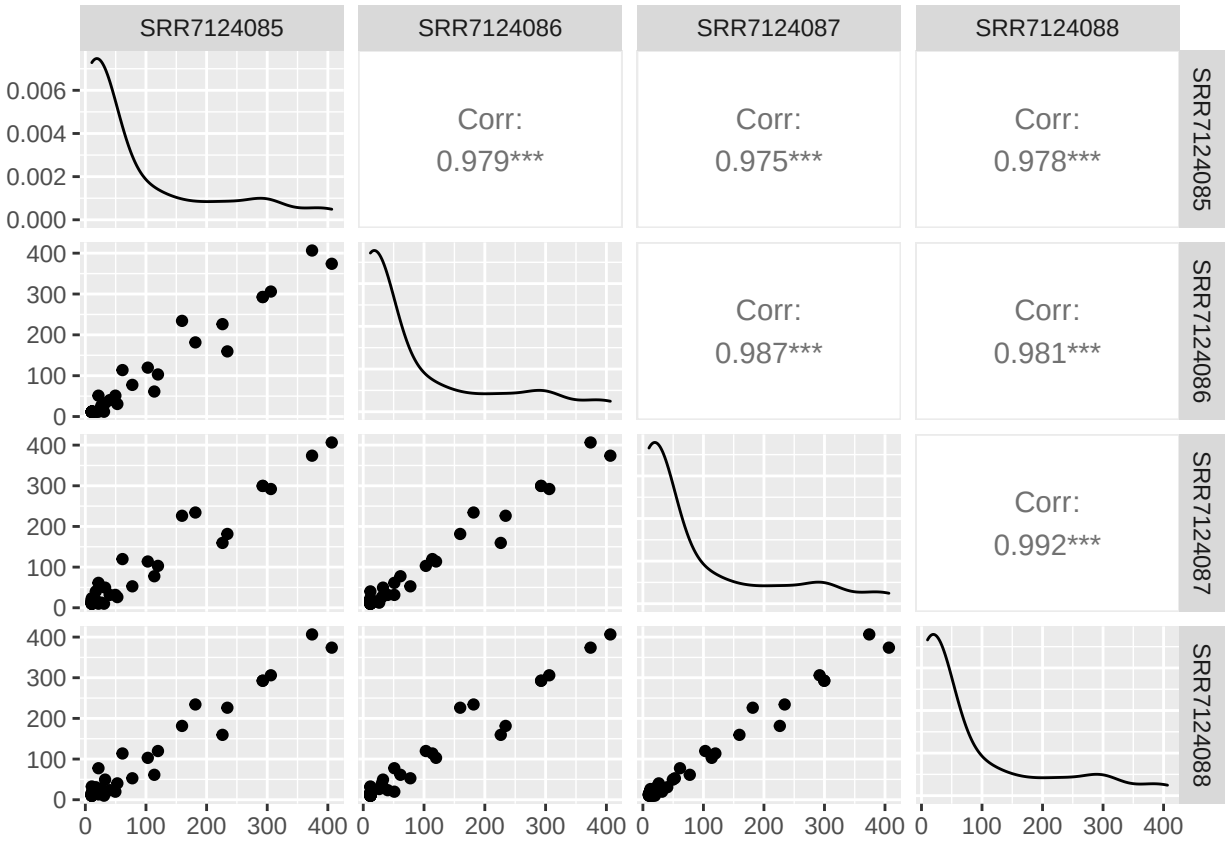


ggpairs

```
ggpairs(cts.qnorm.filtered[rowSums(cts.qnorm.filtered)<2000,])
```

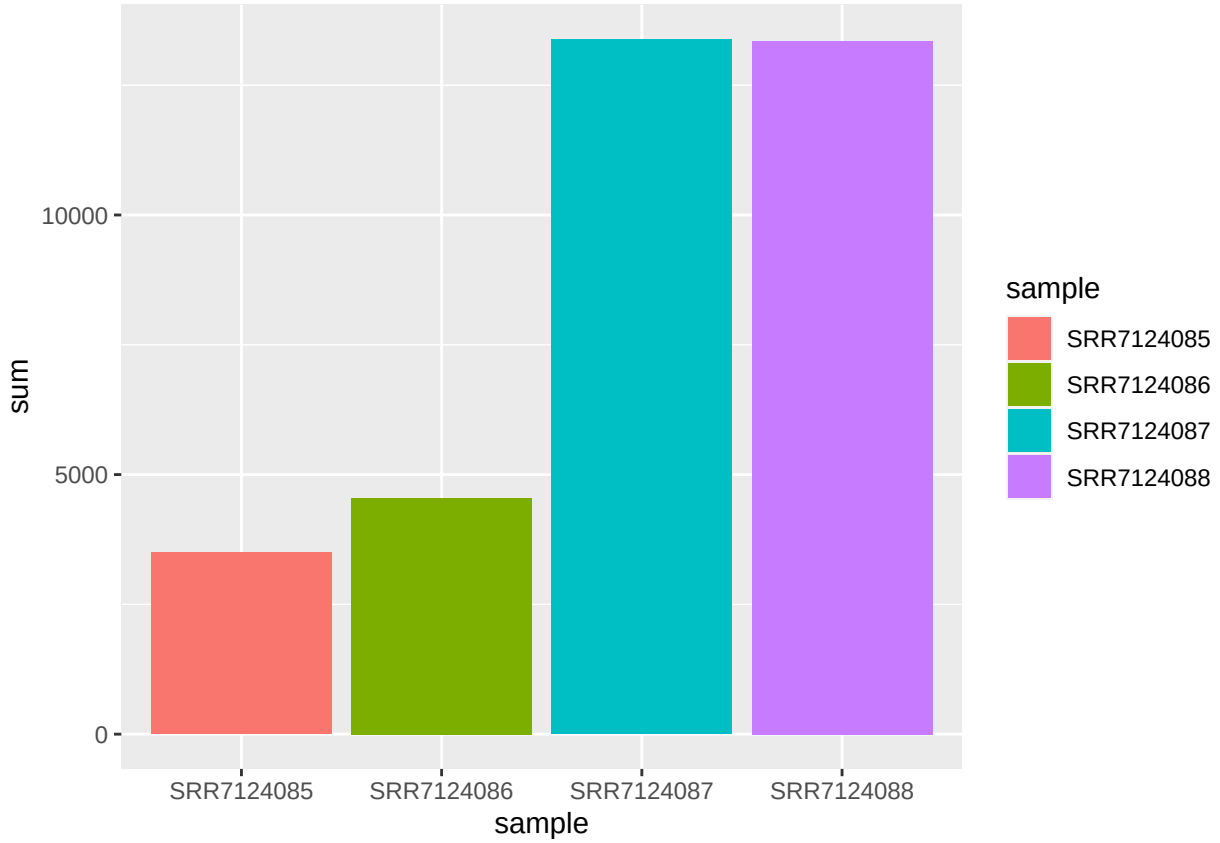



```
cts.filtered = cts.filtered[rowSums(cts.filtered)<2000,]
cts.qnorm.filtered=data.frame(normalize.quantiles(as.matrix(cts.filtered)),
                              row.names=rownames(cts.filtered))
colnames(cts.qnorm.filtered)=colnames(cts.filtered)
ggpairs(cts.qnorm.filtered)
```



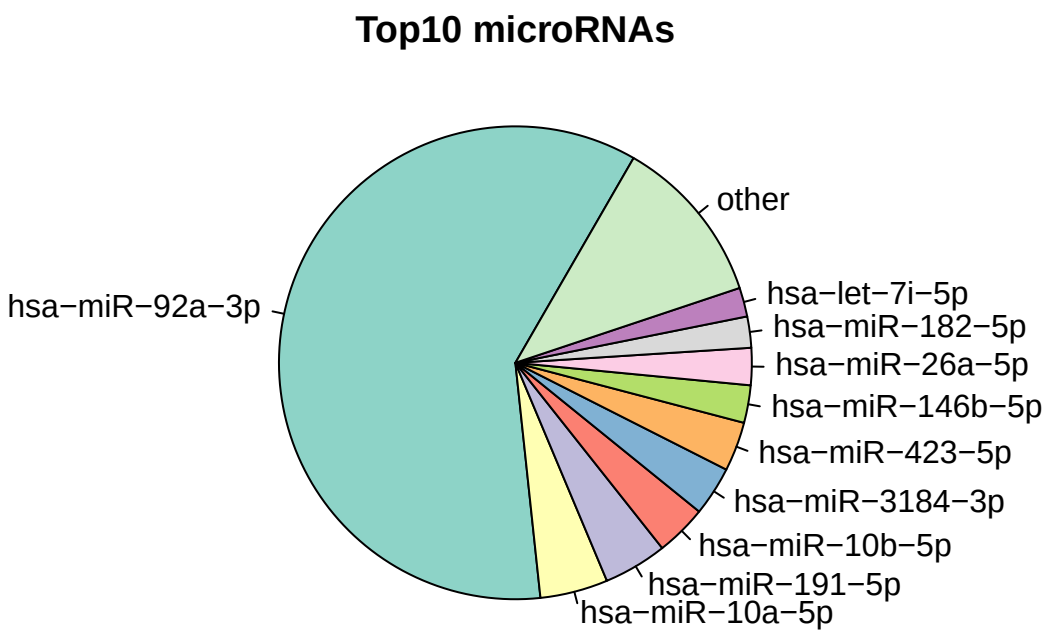
Sum barplot

```
cts.sum = data.frame(sum=(colSums(cts)))
cts.sum$sample = rownames(cts.sum)
ggplot(cts.sum,aes(x=sample,y=sum,fill=sample))+geom_bar(stat="identity")
```

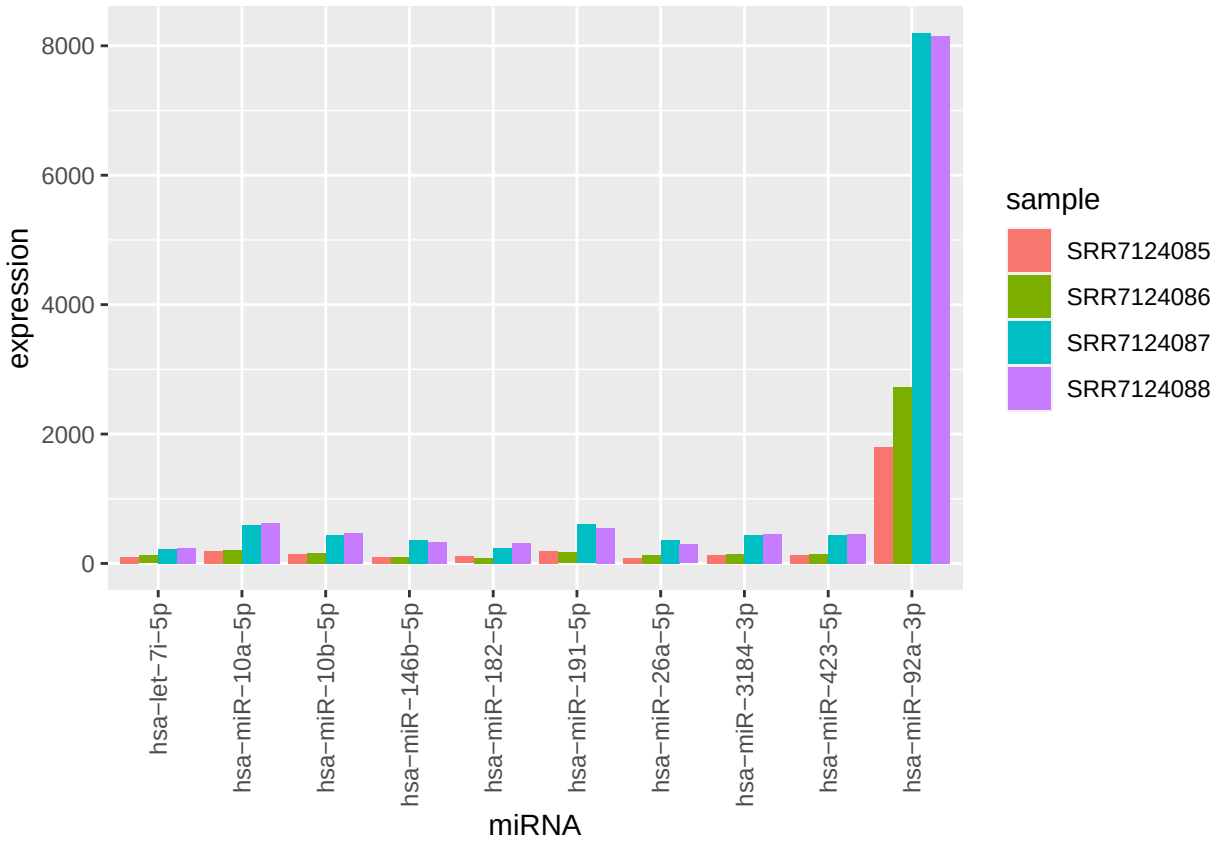


Visualing highest abundance miRNAs

```
cts = cts[order(-rowSums(cts)),]
top10=apply(cts,1,sum)[1:10]
top10[11]=sum(apply(cts,1,sum)[11:nrow(cts)])
names(top10)[11]="other"
pie(top10,col=brewer.pal(11,"Set3"),
    main="Top10 microRNAs",radius=1,init.angle=60)
```




```
cts.top10 = head(cts,10)
cts.tidy = pivot_longer(cts.top10, cols=colnames(cts.top10),
                        names_to='sample', values_to='expression')
cts.tidy$expression[cts.tidy$expression == 0] = NA
mirnalist = c()
for (i in rownames(cts.top10)){mirnalist = c(mirnalist,rep(i,4))}
cts.tidy$miRNA = mirnalist
ggplot(cts.tidy,aes(x=miRNA,y=expression,fill=sample))+
  geom_bar(stat="identity", position=position_dodge())+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

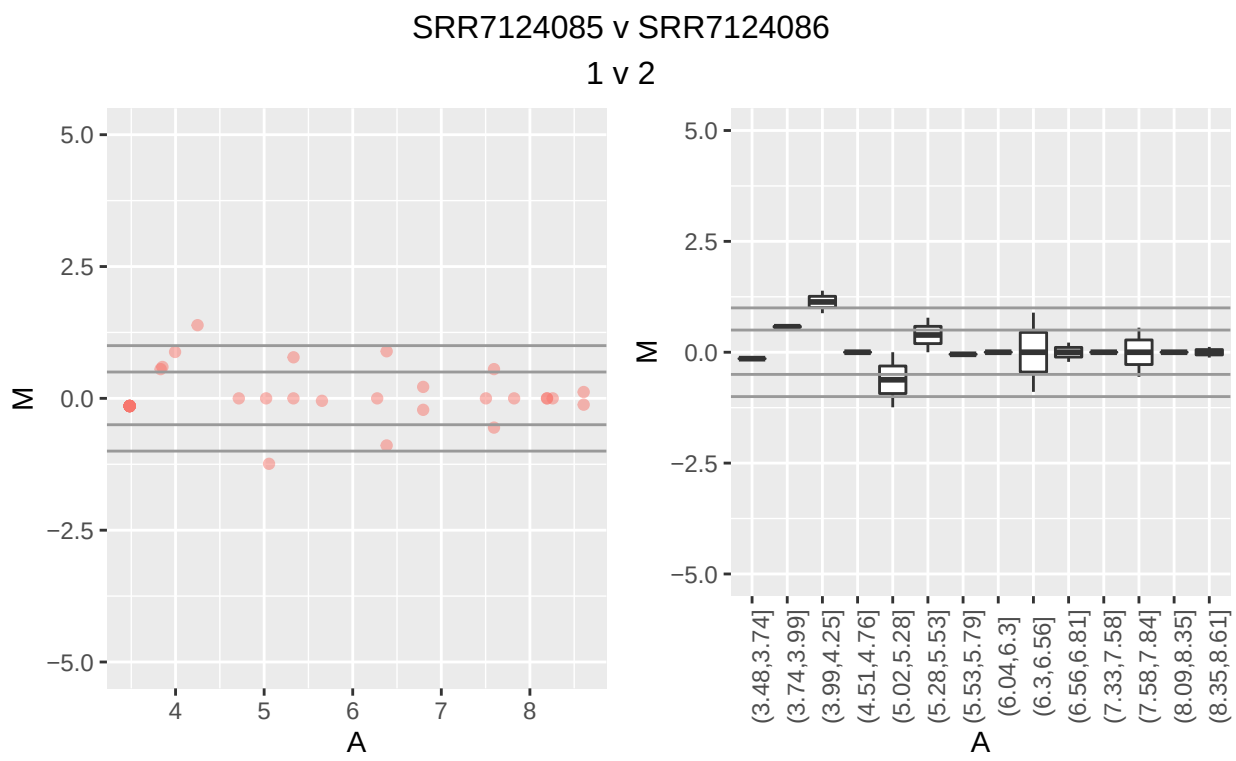


MA plots

```
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$cell[i], 'v',
                    meta$cell[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=0.5)+
    theme(legend.position = "none") +
    geom_hline(yintercept=0.5,colour='gray60')+
    geom_hline(yintercept=-0.5,colour='gray60')+
```

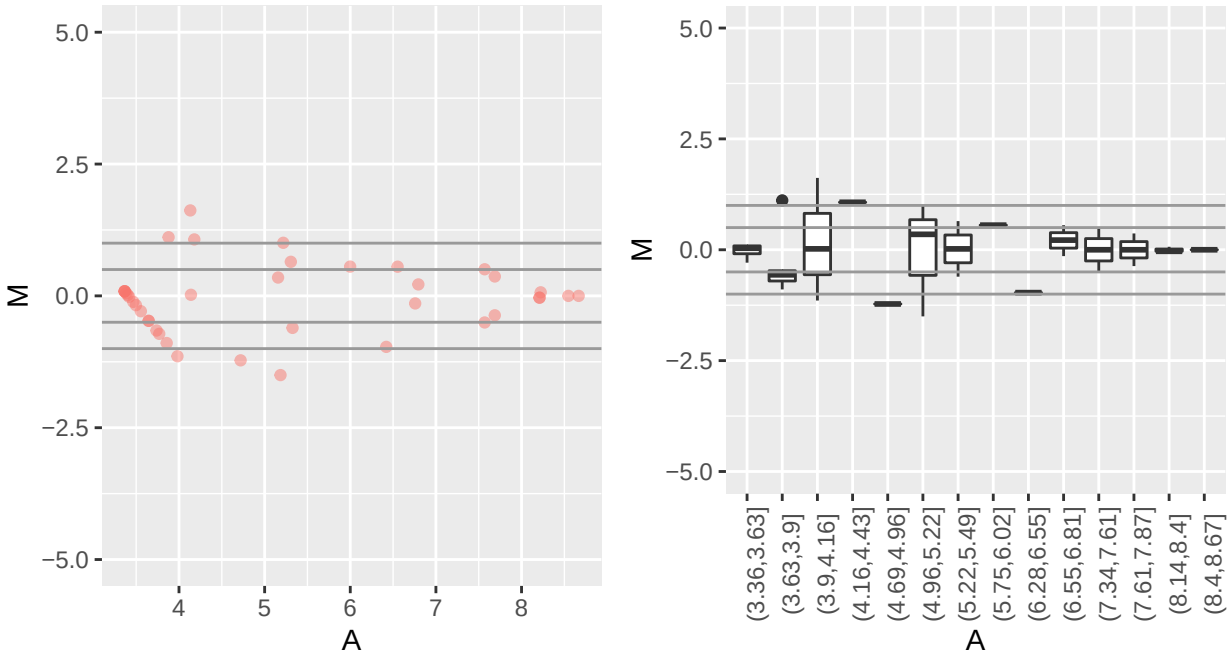
```
    geom_hline(yintercept=1,colour='gray60')+
    geom_hline(yintercept=-1,colour='gray60')
  a.binned = cut(a, 20)
  data.binned = data.frame(A = a.binned, M = m)
  q = ggplot(data=data.binned) +
    geom_boxplot(aes(A, M)) +
    theme(axis.text.x=element_text(angle=90))+
    theme(legend.position = "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 1:3){
  for (j in ((i+1):4))
    ma.plot(cts.qnorm.filtered, meta, i, j, llim, ulim)
  }
```



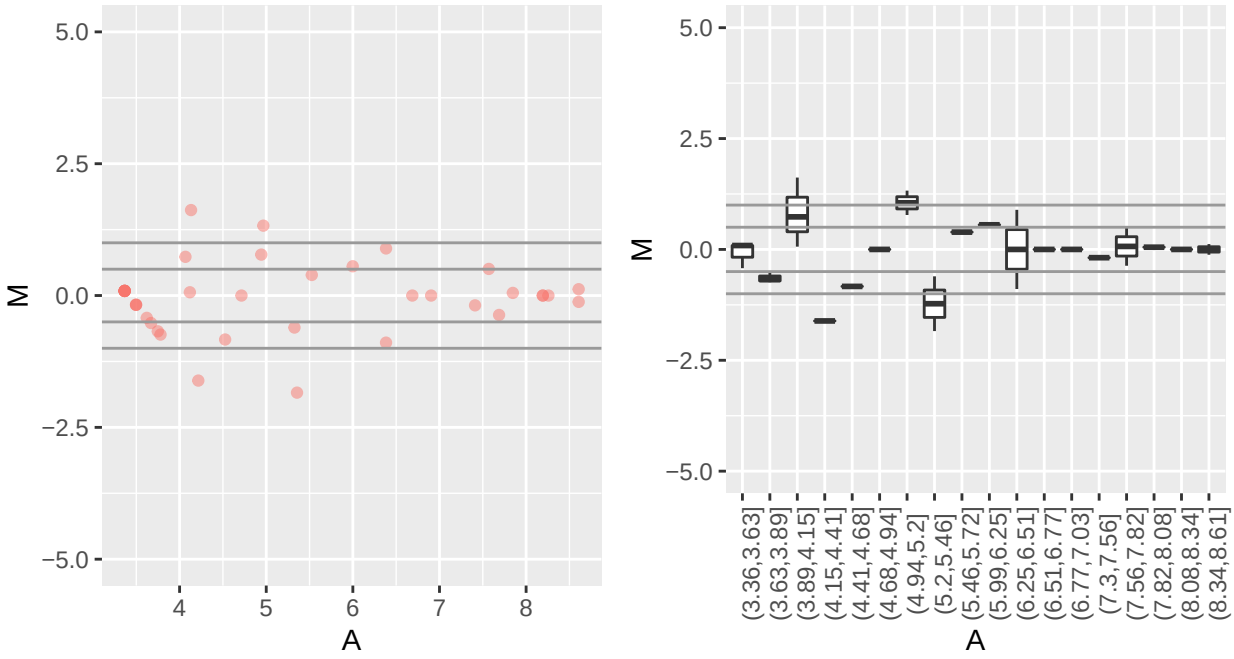
SRR7124085 v SRR7124087

1 v 3



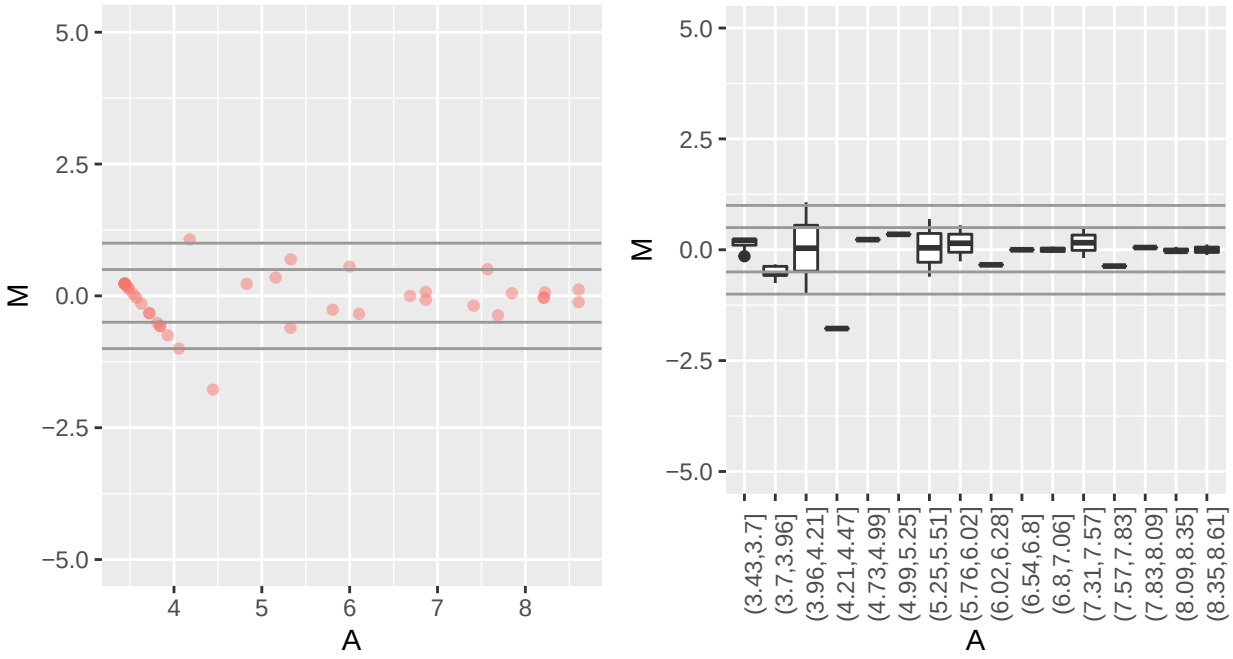
SRR7124085 v SRR7124088

1 v 4



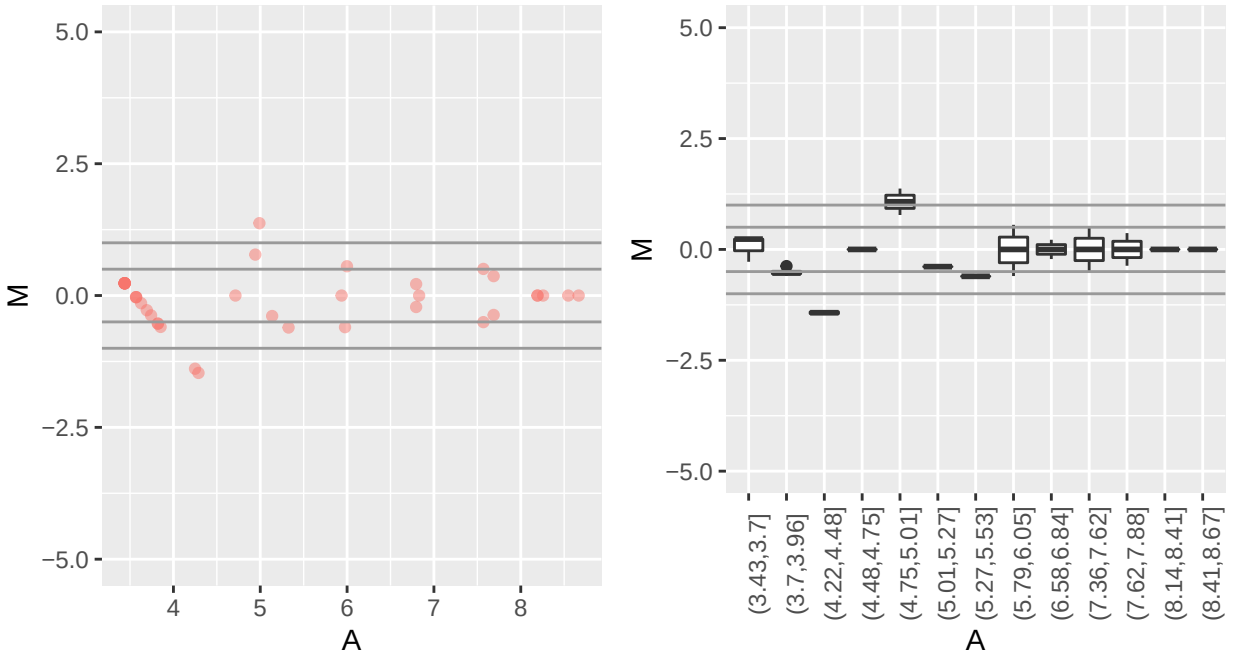
SRR7124086 v SRR7124087

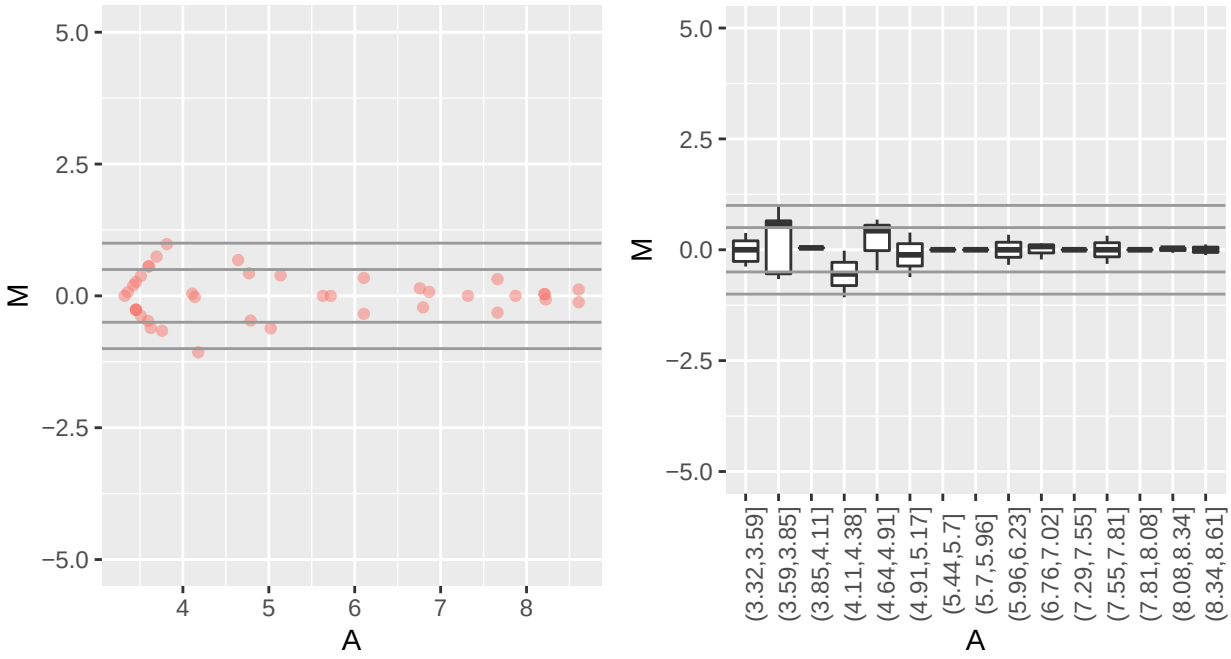
2 v 3



SRR7124086 v SRR7124088

2 v 4





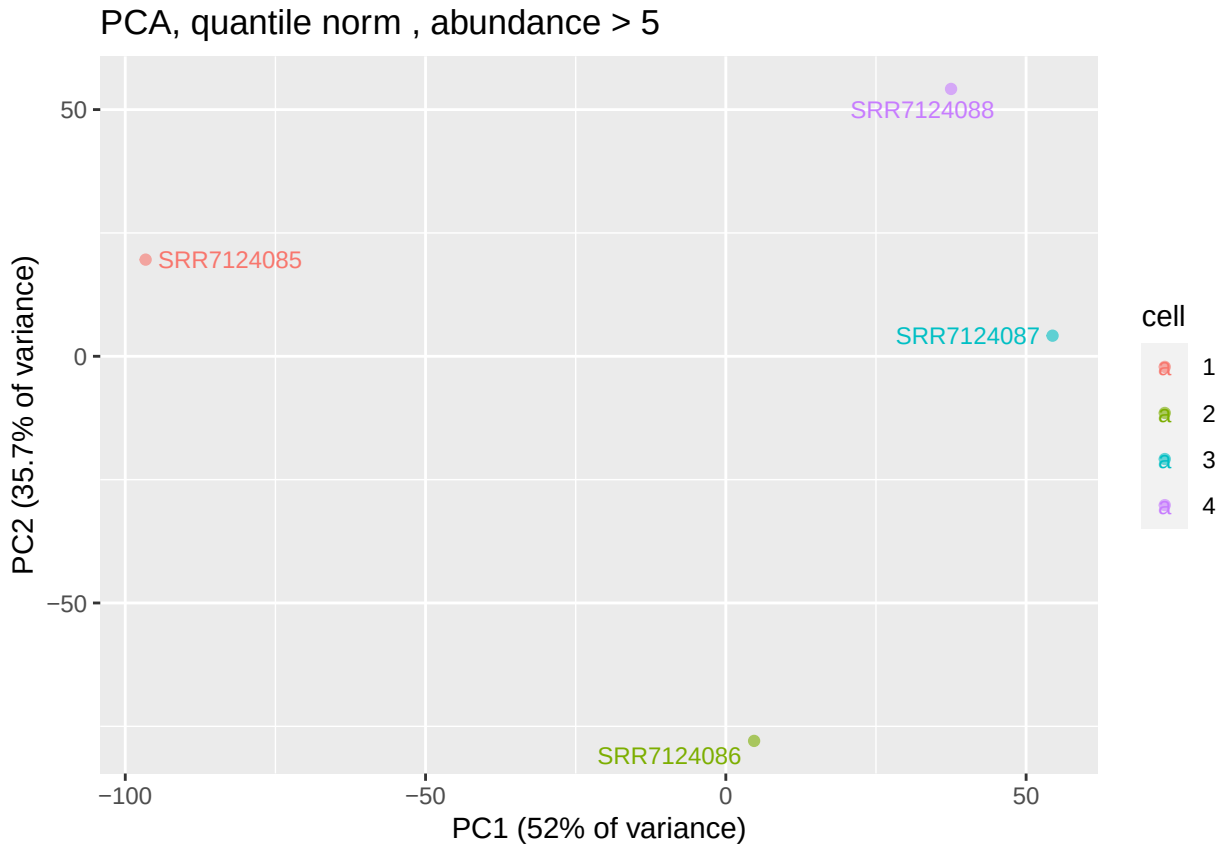
PCA

```
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=cell)) +
    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(seq(1,20))){
  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=5
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

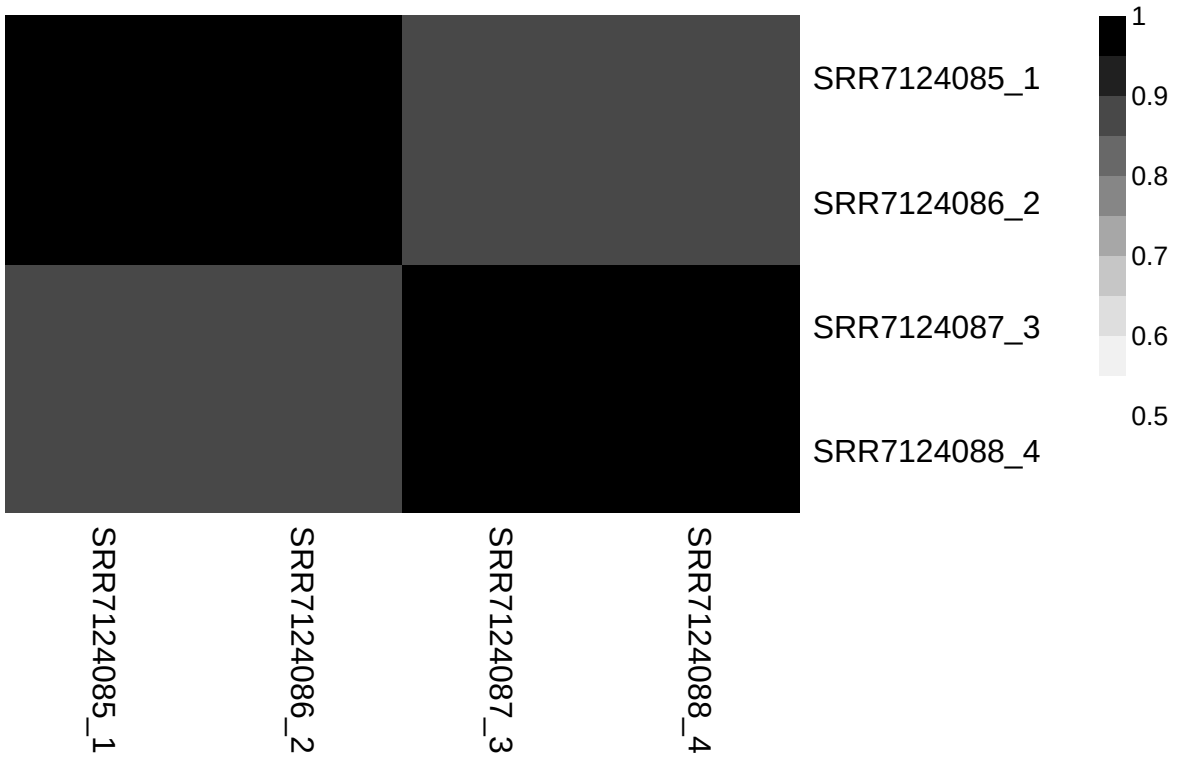
```
jaccard.index = function(a, b){
  if ((length(a) == 0) & (length(b) == 0)){
    return(1)
  } else{
    u = length(union(a,b))
    i = length(intersect(a,b))
    return(i/u)
  }
```

```
}

leaf.labels=paste(meta$id,meta$cell,sep='_')
jaccard.heatmap = function(counts, n.abundant, labels){
  colnames_counts=paste0(meta$id,meta$cell)
  labels=labels[order(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(50,40,30,20,10,5)
for (n in n.abundances){
  jaccard.heatmap(cts, n, leaf.labels)
}
dev.off()
```

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

```
jaccard.heatmap(cts, 15, leaf.labels)
```


Jaccard index of 15 most abundant genes
SRR7124085_1
SRR7124086_2
SRR7124087_3
SRR7124088_4
SRR7124085_1
SRR7124086_2
SRR7124087_3
SRR7124088_4
1
0.9
0.8
0.7
0.6
0.5