DE_polyA_vs_pseudo

Ariel Hippen

2022-07-31

Contents

DESeq2 pipeline

As a control for our DE analysis, we will run differential expression on our real bulk results and pseudo-bulk data (single-cell data pooled together to approximate a bulk RNA-seq sample.) Here we'll compare to our dissociated, poly-A captured bulk data. Since that's the same library prep process, we'd hope that there's not a ton of significant changes here, but let's be honest we're probably not that lucky.

```
suppressPackageStartupMessages({
  library(data.table)
  library(DESeq2)
  library(vsn)
  library(pheatmap)
  library(RColorBrewer)
  library(PCAtools)
  library(testit)
  library(biomaRt)
  library(dplyr)
  library(yaml)
  library(cqn)
})

params <- read_yaml("../../config.yml")
data_path <- params$data_path
local_data_path <- params$local_data_path
samples <- params$samples
```

Load data

```
# Get paths to STAR.counts files for bulk poly-A samples
# Note: variable data_path is loaded from config.R
directory <- paste(data_path, "bulk_tumors", sep = "/")
sampleFiles <- list.files(directory, recursive = TRUE, full.names = TRUE)
sampleFiles <- grep("ReadsPerGene.out.tab", sampleFiles, value=TRUE)
sampleFiles <- grep("polyA",sampleFiles, value=TRUE)
```

DESeq expects a metadata table to pass into the colData part of a SummarizedExperiment object. We'll prep it here.

```
# Note: variable samples is loaded from config.R
sampleNames <- c(gsub(".*/(\\d+)/.*", "\\1", sampleFiles), samples)
sampleCondition <- rep(c("bulk","pseudo"), each=8)
samplePool <- ifelse(sampleNames %in% c("2251","2428","2467","2497"),
                     "01132022","12162021")
sampleUnique <- paste(sampleNames,sampleCondition,sep="_")
colData <- data.frame(id = sampleUnique,
                      sample = sampleNames,
                      pool = samplePool,
                      condition = sampleCondition)
colData$condition <- factor(colData$condition)
```

Now we can load in the bulk files and create a counts matrix. Note that this shouldn't be stranded data, so we'll use column 2 of the STAR counts files.

```
counts <- matrix(nrow=36601, ncol = 8)
for (i in 1:8){
  newcounts <- fread(sampleFiles[i])
  newcounts <- newcounts[-c(1:4),]
  counts[, i] <- newcounts$V2
  if (i == 1) {
    rownames(counts) <- newcounts$V1
  } else{
      assert(rownames(counts) == newcounts$V1)
  }
}
rm(newcounts); gc()
```

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  8284176 442.5   14530478 776.1 11060729 590.8
## Vcells 14552435 111.1   22157933 169.1 18394002 140.4
```

And now we'll add in the pseudobulk data from generate_pseudobulk.R.

```
sce_path <- paste(local_data_path, "sce_objects", sep = "/")
pseudo <- readRDS(paste(sce_path, "full_pseudobulk.rds", sep = "/"))
counts <- cbind(counts, pseudo)
colnames(counts) <- colData$id
rm(pseudo); gc()
```

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  8284875 442.5   14530478 776.1 11060729 590.8
## Vcells 14974312 114.3   26669519 203.5 18394002 140.4
```

We'll use gene length and GC content to do CQN normalization. First pull the info from biomaRt:

```
mart <- useEnsembl("ensembl", dataset = "hsapiens_gene_ensembl")
gene_coords=getBM(attributes=c("hgnc_symbol","ensembl_gene_id","start_position","end_position",
                               "percentage_gene_gc_content"), filters="ensembl_gene_id",
                  values=rownames(counts), mart=mart)
gene_coords$size=gene_coords$end_position - gene_coords$start_position
head(gene_coords)
```

```
##   hgnc_symbol ensembl_gene_id start_position end_position
## 1       SCYL3 ENSG00000000457      169849631    169894267
## 2    C1orf112 ENSG00000000460      169662007    169854080
## 3         FGR ENSG00000000938       27612064     27635185
## 4         CFH ENSG00000000971      196651754    196752476
## 5       STPG1 ENSG00000001460       24356999     24416934
## 6      NIPAL3 ENSG00000001461       24415802     24472976
##   percentage_gene_gc_content   size
## 1                      40.14  44636
## 2                      39.22 192073
## 3                      52.92  23121
## 4                      35.08 100722
## 5                      44.09  59935
## 6                      44.99  57174
```

Subset counts to genes with length info and check they're ordered properly:

```
gene_coords <- subset(gene_coords, gene_coords$ensembl_gene_id %in% rownames(counts))
gene_coords <- gene_coords[!duplicated(gene_coords$ensembl_gene_id),]
counts <- counts[rownames(counts) %in% gene_coords$ensembl_gene_id,]

counts <- counts[order(rownames(counts)),]
gene_coords <- gene_coords[order(gene_coords$ensembl_gene_id),]

stopifnot(rownames(counts)==gene_coords$ensembl_gene_id)
```

```
cqn <- cqn(counts = counts,
         x = gene_coords$percentage_gene_gc_content,
         lengths = gene_coords$size)

cqnOffset <- cqn$glm.offset
cqnNormFactors <- exp(cqnOffset)
cqnNormFactors <- cqnNormFactors / exp(rowMeans(log(cqnNormFactors)))
```

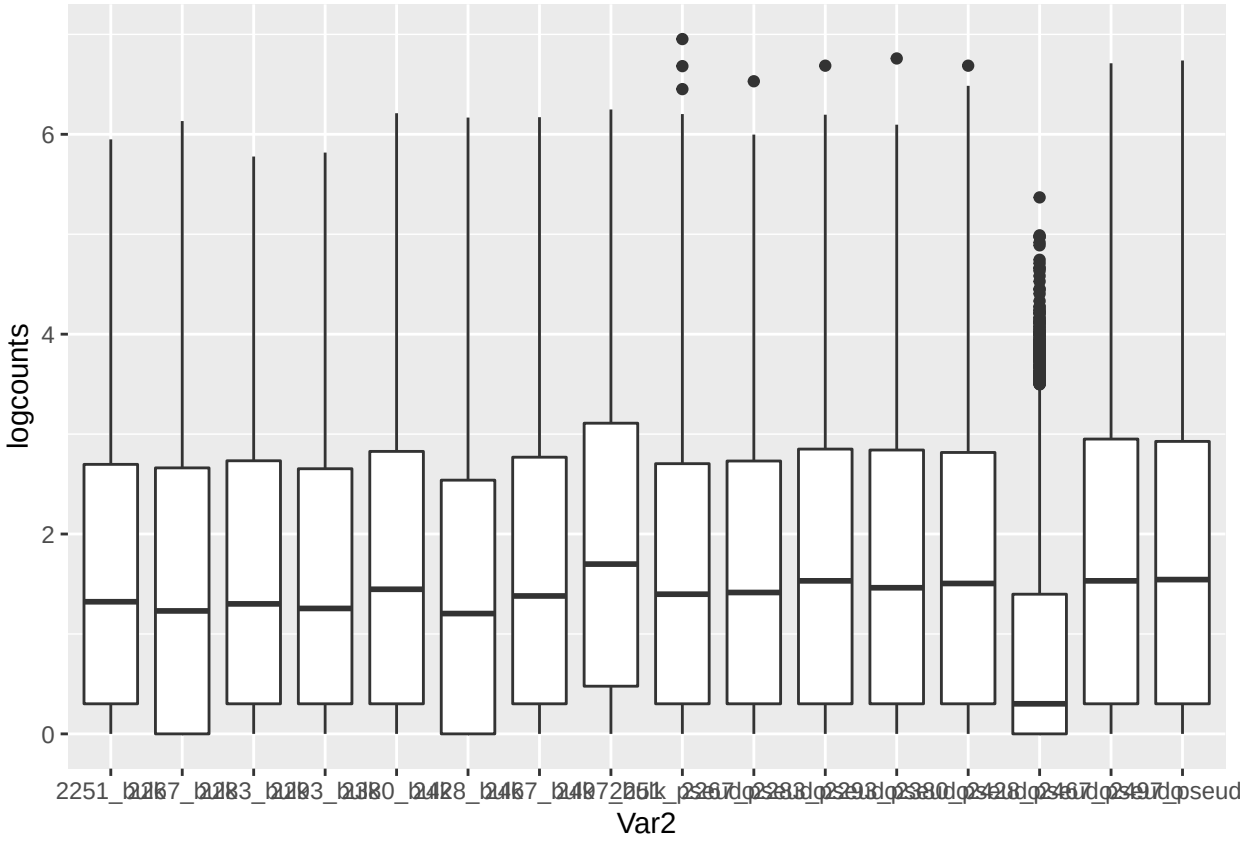
This counts matrix has the genes listed by their Ensembl IDs, which is helpful for uniqueness but bad for readability in the downstream analysis. The easiest way I've found to convert ensembl IDs to gene names for this dataset is by loading in a SingleCellExperiment object from the same experiment, where this mapping is automatically stored.

```
#sce <- readRDS(paste(sce_path, "pooled_clustered_50.rds", sep = "/"))
#gene_map <- as.data.frame(subset(rowData(sce), select=c("ID","Symbol")))
#gene_map <- gene_map[rownames(counts)==gene_map$ID,]
#rownames(counts) <- gene_map$Symbol
#rm(sce); gc()
```

```
# Create DESeq object
dds <- DESeqDataSetFromMatrix(countData = counts,
                              colData = colData,
                              design = ~ sample + condition)
normalizationFactors(dds) <- cqnNormFactors
```

The tutorial recommends doing some basic pre-filtering of non- or low-expressed genes to speed up computation, estimate the library depth correction factor, and to clean up later visualizations.

```
colnames(counts) <- colData$id
melted_counts <- melt(counts)
melted_counts$logcounts <- log10(melted_counts$value+1)

ggplot(melted_counts, aes(x=Var2, y=logcounts)) + geom_boxplot()
```



Okay, in some of the samples at least 25% of the genes are not expressed. Let's go a little more conservative and remove any genes that have fewer than 20 reads total.

```
keep <- rowSums(counts(dds)) >= 20
dds <- dds[keep, ]
```

Differential expression

If you don't set the condition factor specifically, it can be hard to tell if A is upregulated compared to B or vice versa. We'll set "chunk_ribo" as the reference and look at how dissociated_ribo is upregulated or downregulated compared to that.

```
dds$condition <- relevel(dds$condition, ref = "bulk")
```

```
# Run differential expression
dds <- DESeq(dds)
res <- results(dds)
res
```

```
## log2 fold change (MLE): condition pseudo vs bulk
## Wald test p-value: condition pseudo vs bulk
## DataFrame with 28931 rows and 6 columns
##                   baseMean log2FoldChange     lfcSE       stat      pvalue
##                  <numeric>      <numeric> <numeric>  <numeric>   <numeric>
## ENSG00000000003 1431.2316     -0.8152110  0.259912  -3.136483 1.70987e-03
## ENSG00000000005   13.2279     -0.5921824  0.521407  -1.135739 2.56066e-01
## ENSG00000000419 2185.5728      0.0255513  0.159683   0.160013 8.72871e-01
## ENSG00000000457  288.0255     -0.8173131  0.120511  -6.782084 1.18454e-11
## ENSG00000000460  234.2061     -0.5003259  0.185262  -2.700646 6.92049e-03
## ...                   ...            ...       ...        ...         ...
## ENSG00000288235 15.544982      -2.023159  0.624479  -3.239757 1.19632e-03
## ENSG00000288253 34.732485      -0.669937  0.315853  -2.121037 3.39187e-02
## ENSG00000288302 27.389299       3.820505  0.498455   7.664687 1.79268e-14
## ENSG00000288321  0.710415       0.860798  3.037127   0.283425 7.76851e-01
## ENSG00000288398 92.436935     -2.012745  0.186913 -10.768371 4.85517e-27
##                        padj
##                   <numeric>
## ENSG00000000003 3.37023e-03
## ENSG00000000005 3.21887e-01
## ENSG00000000419 8.99869e-01
## ENSG00000000457 5.41990e-11
## ENSG00000000460 1.23385e-02
## ...                     ...
## ENSG00000288235 2.41543e-03
## ENSG00000288253 5.29203e-02
## ENSG00000288302 1.02316e-13
## ENSG00000288321 8.20767e-01
## ENSG00000288398 5.66848e-26
```

The tutorial says "shrinkage of effect size (LFC estimates) is useful for visualization and ranking of genes. To shrink the LFC, we pass the dds object to the function lfcShrink. We provide the dds object and the name or number of the coefficient we want to shrink."

```
resLFC <- lfcShrink(dds, coef="condition_pseudo_vs_bulk", type="apeglm")
resLFC
```

```
## log2 fold change (MAP): condition pseudo vs bulk
## Wald test p-value: condition pseudo vs bulk
## DataFrame with 28931 rows and 5 columns
##                   baseMean log2FoldChange     lfcSE      pvalue        padj
##                  <numeric>      <numeric> <numeric>   <numeric>   <numeric>
## ENSG00000000003 1431.2316     -0.7740787  0.264117 1.70987e-03 3.37023e-03
## ENSG00000000005   13.2279     -0.4865907  0.500849 2.56066e-01 3.21887e-01
## ENSG00000000419 2185.5728      0.0254536  0.159181 8.72871e-01 8.99869e-01
## ENSG00000000457  288.0255     -0.8087593  0.120533 1.18454e-11 5.41990e-11
## ENSG00000000460  234.2061     -0.4853542  0.185125 6.92049e-03 1.23385e-02
## ...                   ...            ...       ...         ...         ...
## ENSG00000288235 15.544982     -1.7256206  0.703687 1.19632e-03 2.41543e-03
## ENSG00000288253 34.732485     -0.6128849  0.312980 3.39187e-02 5.29203e-02
## ENSG00000288302 27.389299      3.7350356  0.515817 1.79268e-14 1.02316e-13
## ENSG00000288321  0.710415      0.0721202  1.012844 7.76851e-01 8.20767e-01
## ENSG00000288398 92.436935     -1.9885581  0.188102 4.85517e-27 5.66848e-26
```

A quick summary of our differential expression results, at both a 0.1 and 0.05 FDR.

```
summary(res)
```

```
##
## out of 28931 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 8587, 30%
## LFC < 0 (down)     : 11239, 39%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
sum(res$padj  < 0.1, na.rm = TRUE)
```

```
## [1] 19826
```

```
res05 <- results(dds, alpha=0.05)
summary(res05)
```

```
##
## out of 28931 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 7974, 28%
## LFC < 0 (down)     : 10463, 36%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
sum(res05$padj < 0.05, na.rm = TRUE)
```

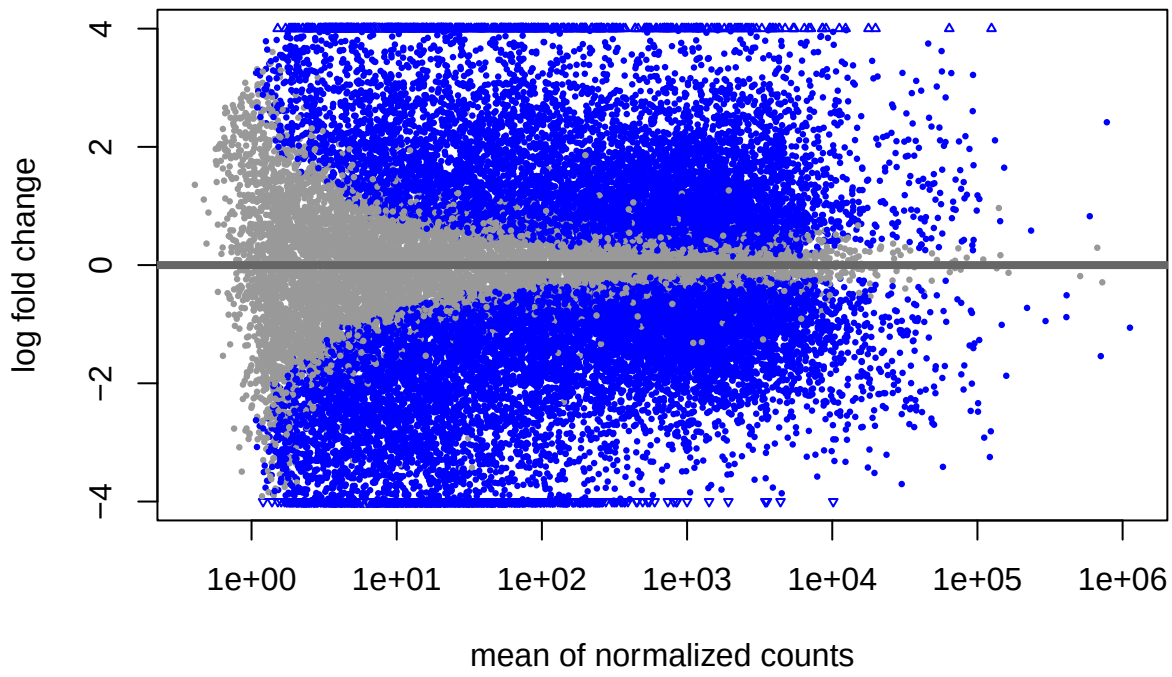
```
## [1] 18437
```

So it seems like there are more genes downregulated than up (meaning there are more genes more highly expressed in true bulk than in pseudobulk). This is what we'd expect, given the technical dropouts of scRNA-seq, but it's worth noting there's still a *lot* of genes upregulated, aka more expressed in pseudobulk than in true bulk.

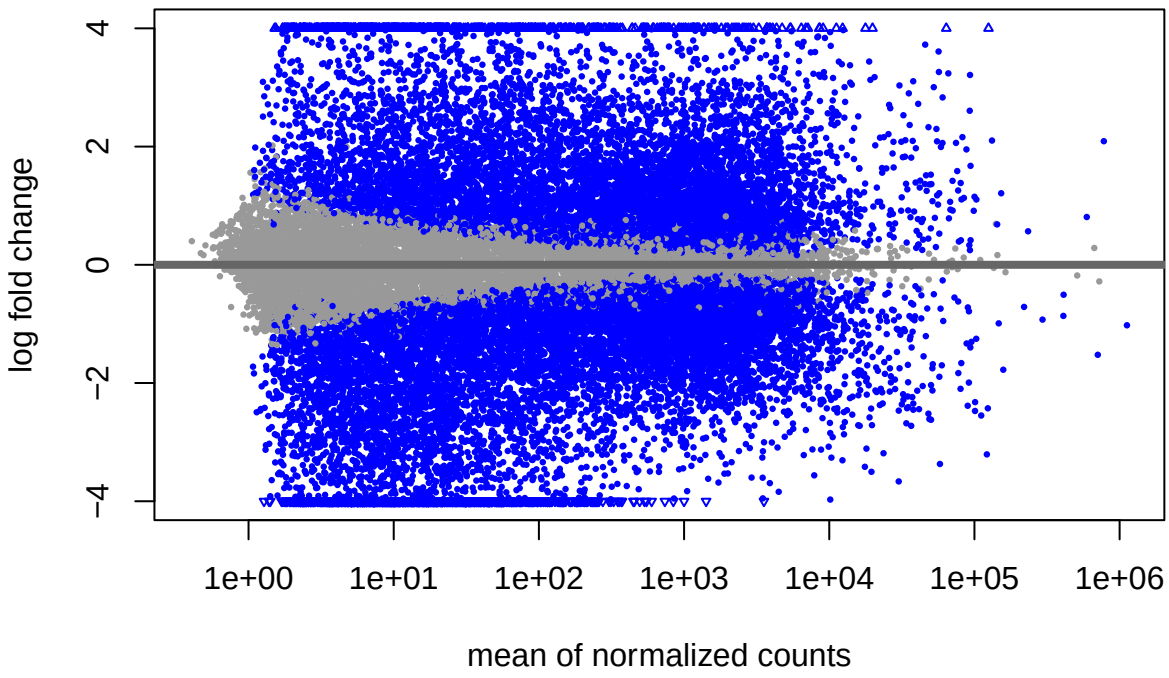
Plotting results

From tutorial: "In DESeq2, the function plotMA shows the log2 fold changes attributable to a given variable over the mean of normalized counts for all the samples in the DESeqDataSet. Points will be colored blue if the adjusted p value is less than 0.1. Points which fall out of the window are plotted as open triangles pointing either up or down."

```
plotMA(res, ylim=c(-4,4))
```




```
plotMA(resLFC, ylim=c(-4,4))
```



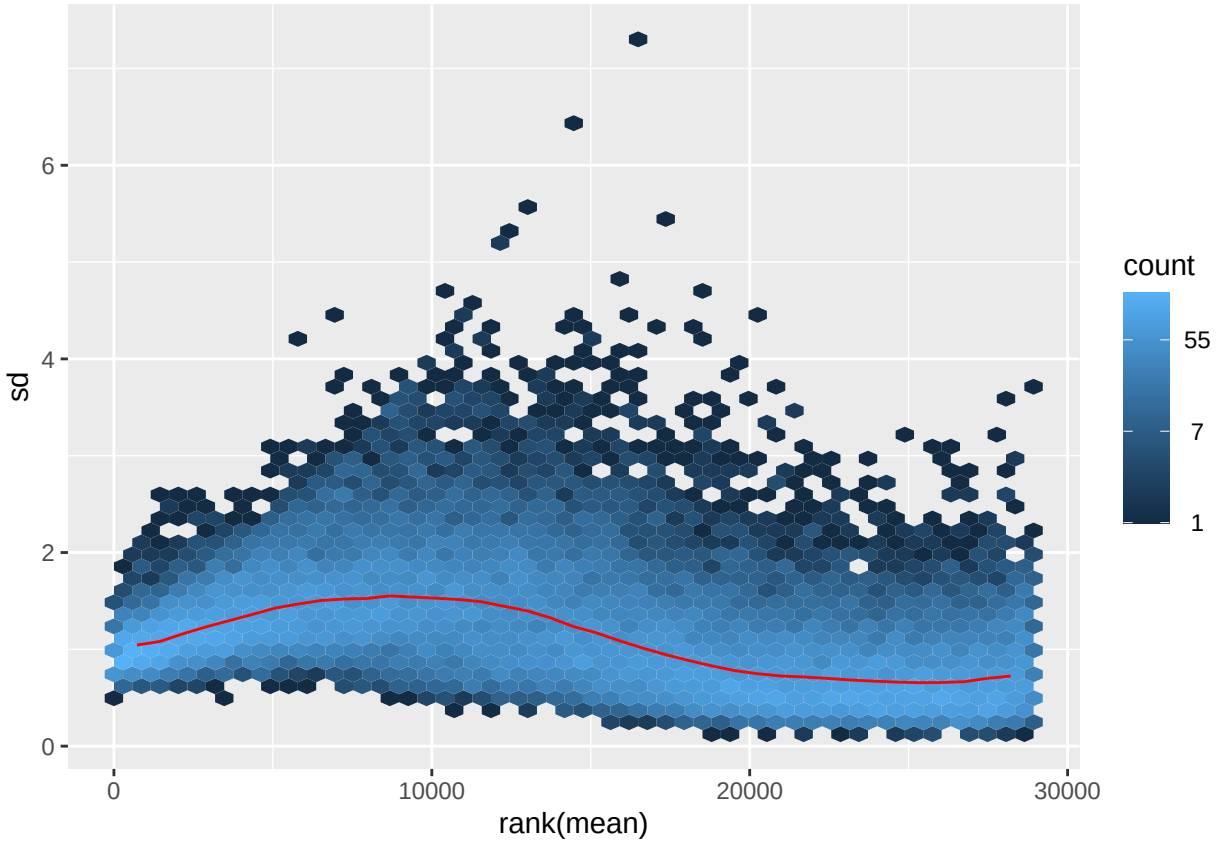
Transformations

The DESeq2 authors recommend the rlog method to adjust for heteroskedasticity in experiments with n < 30. We'll check it and the other vst method they recommend for n > 30.

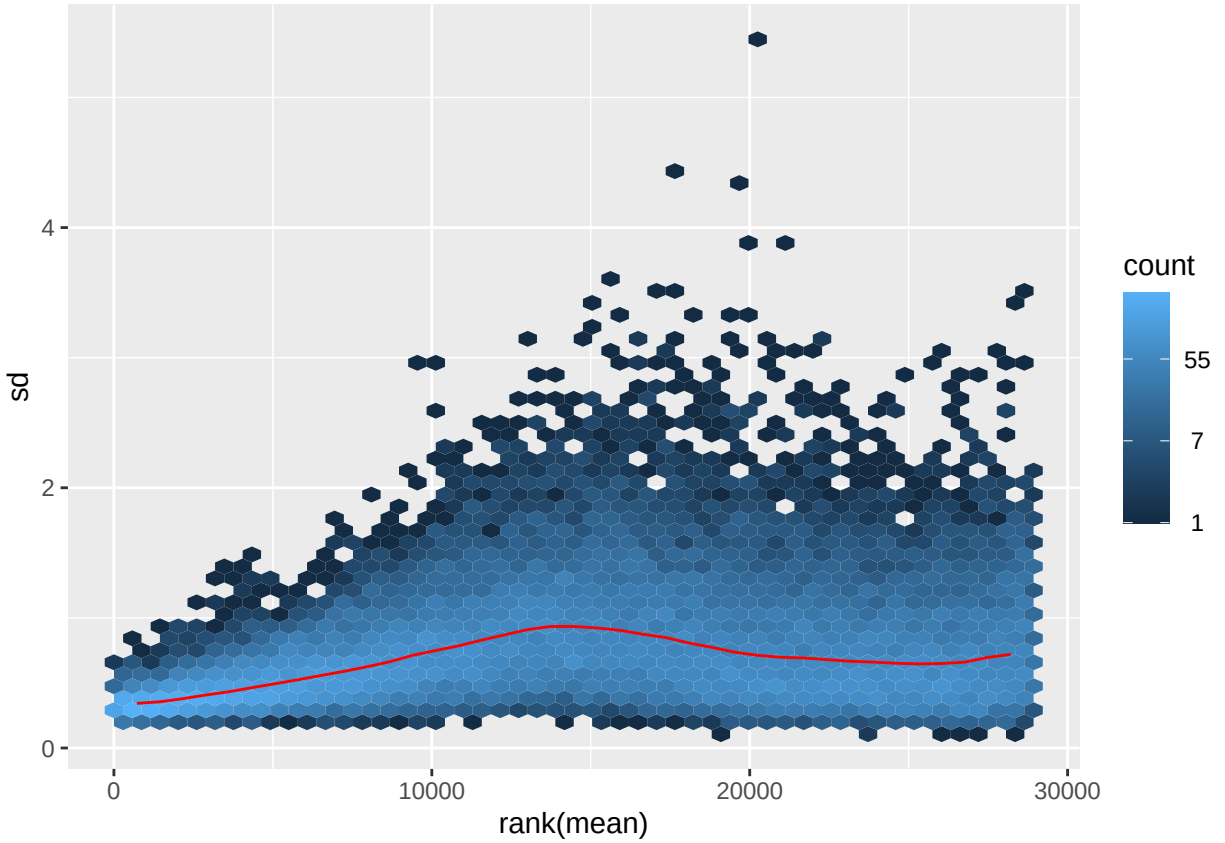
```
vsd <- vst(dds, blind=FALSE)
rld <- rlog(dds, blind=FALSE)
```

The meanSdPlot plots the mean (as ranked values) by standard deviation, if there is heteroskedasticity there should be a flat line across the values, but they say we shouldn't expect it to be perfectly straight.

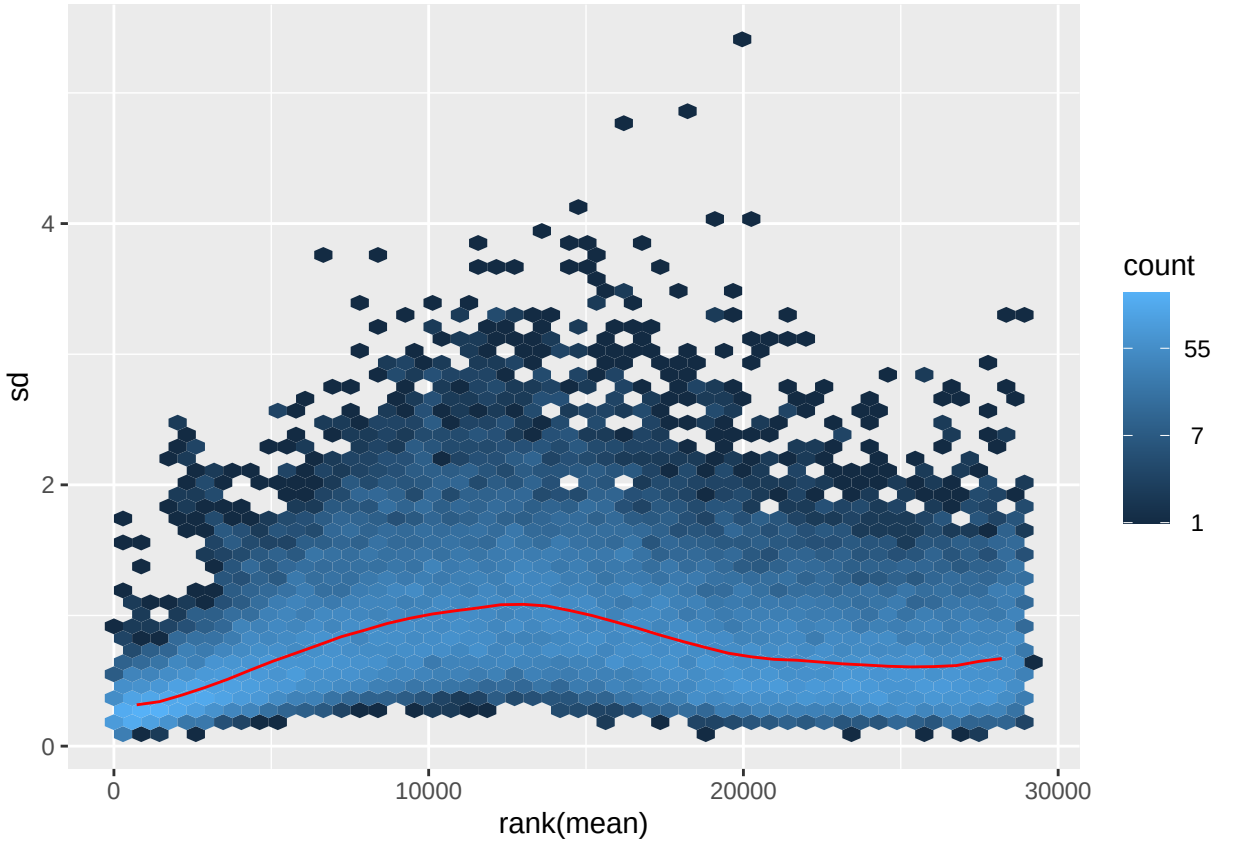
```
ntd <- normTransform(dds)
meanSdPlot(assay(ntd))
```




```
meanSdPlot(assay(vsd))
```



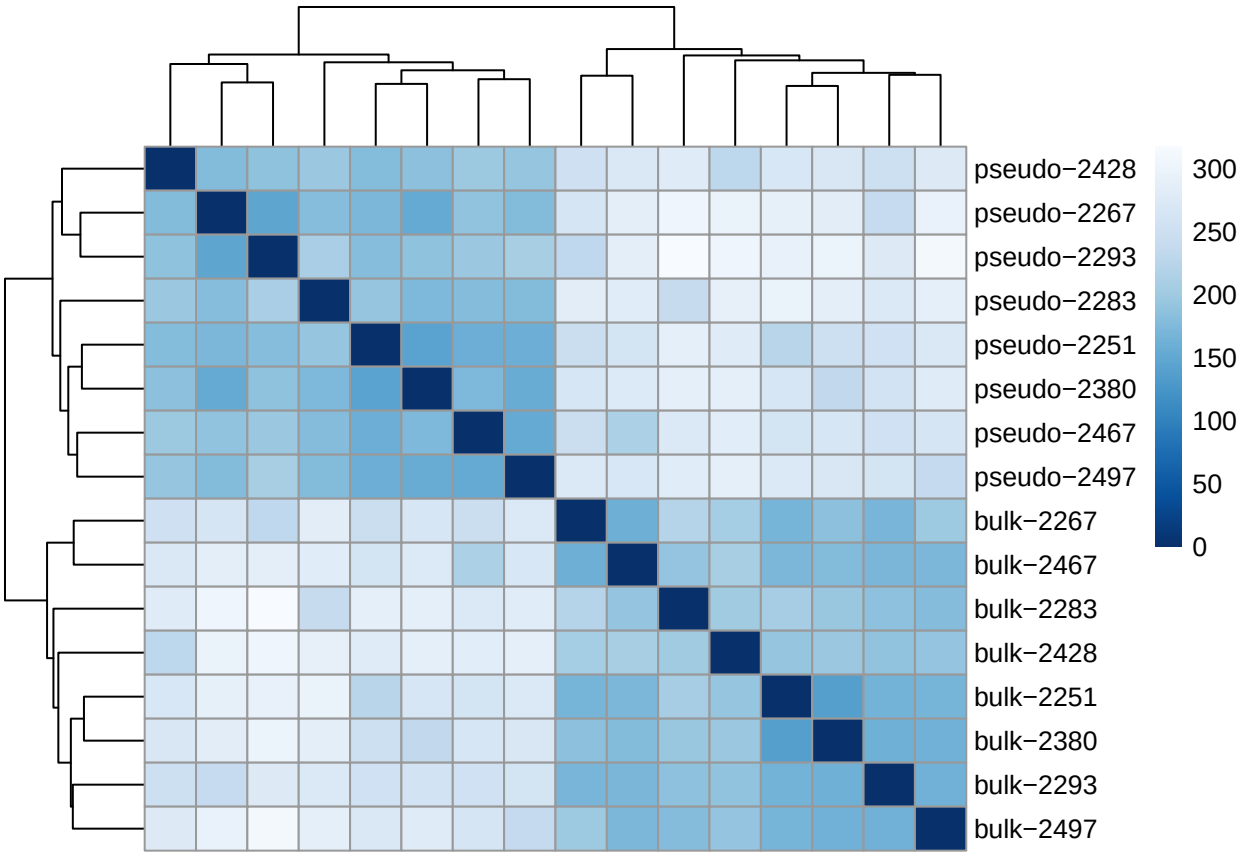

```
meanSdPlot(assay(rld))
```



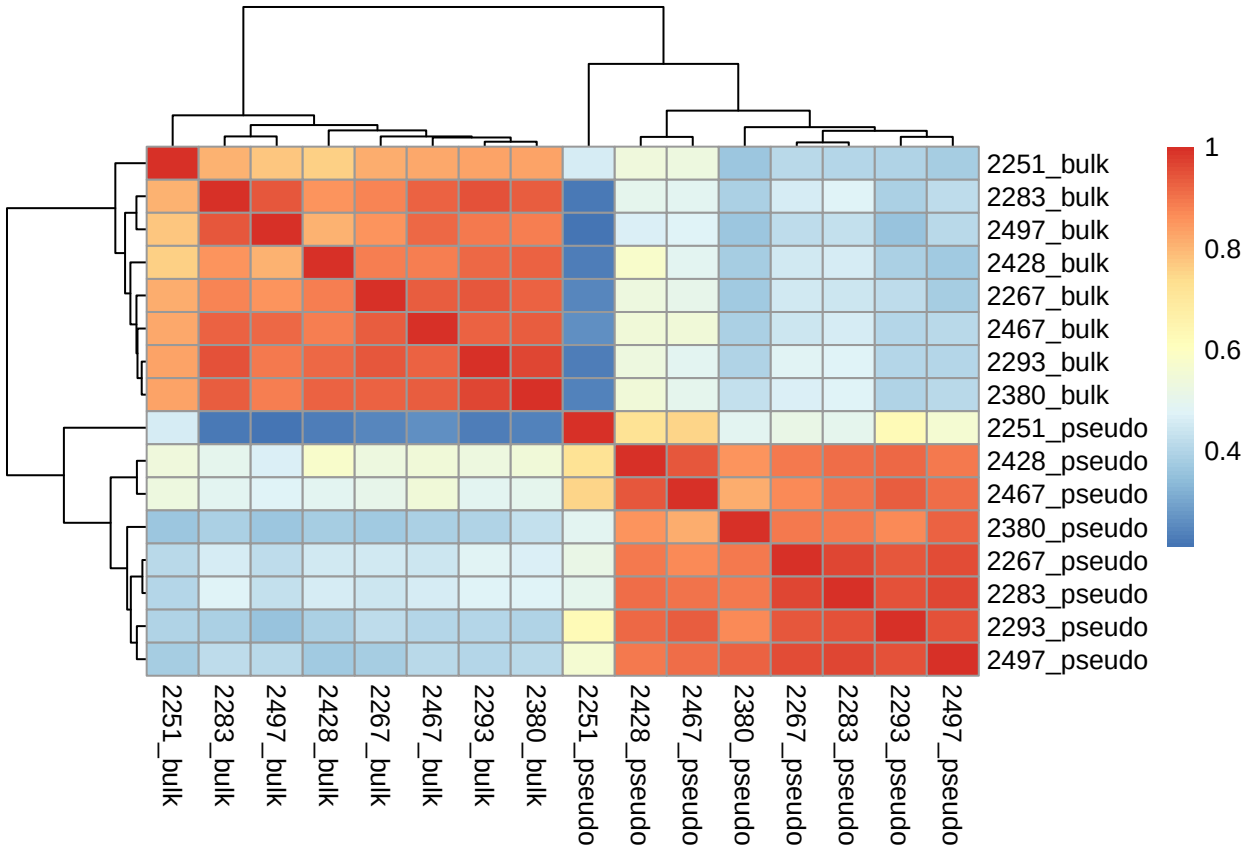
Okay, none of these are *too* heteroskedastic, but the normal and rlog look slightly better than vsd. I'll use rlog for the sample comparisons.

Sample comparisons

```
sampleDists <- dist(t(assay(rld)))

sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(rld$condition, rld$sample, sep="-")
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors)
```

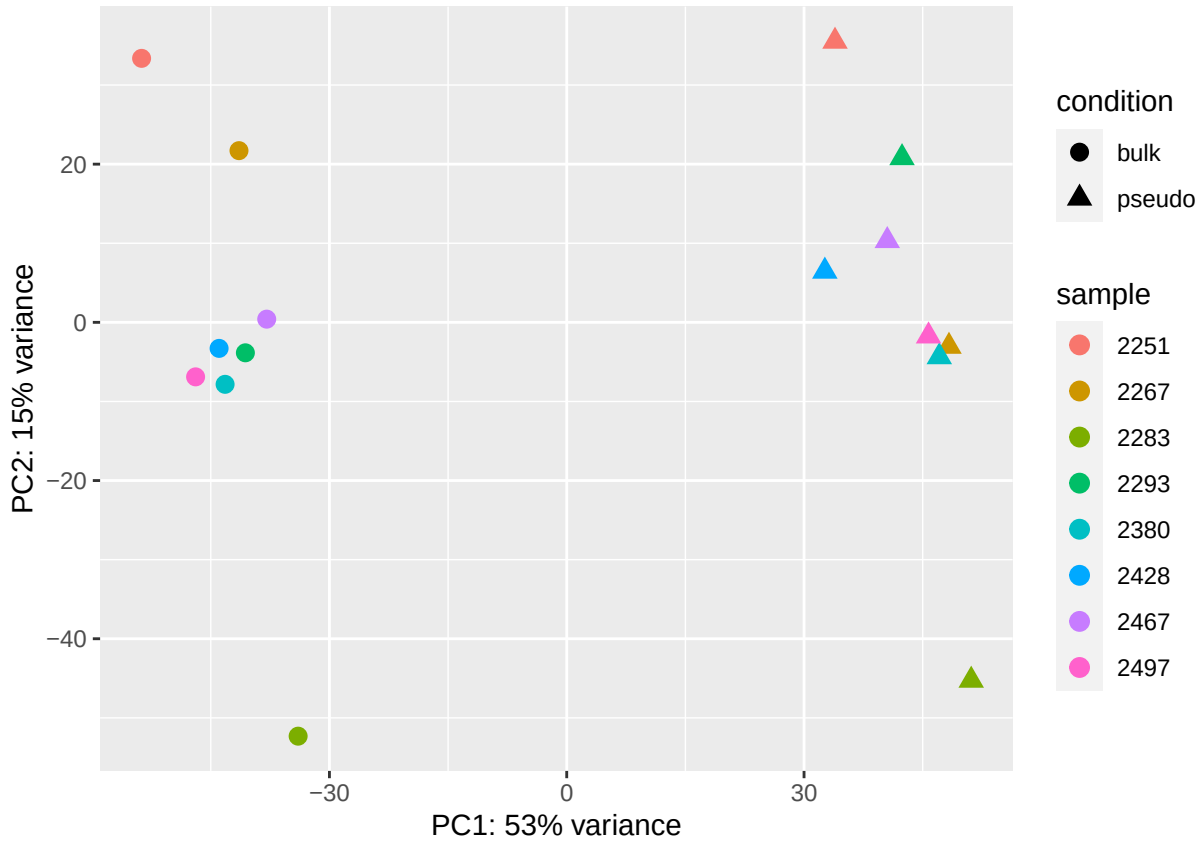



```
x<-cor(counts)
pheatmap(x)
```



The separation is clearly sequencing type-specific.

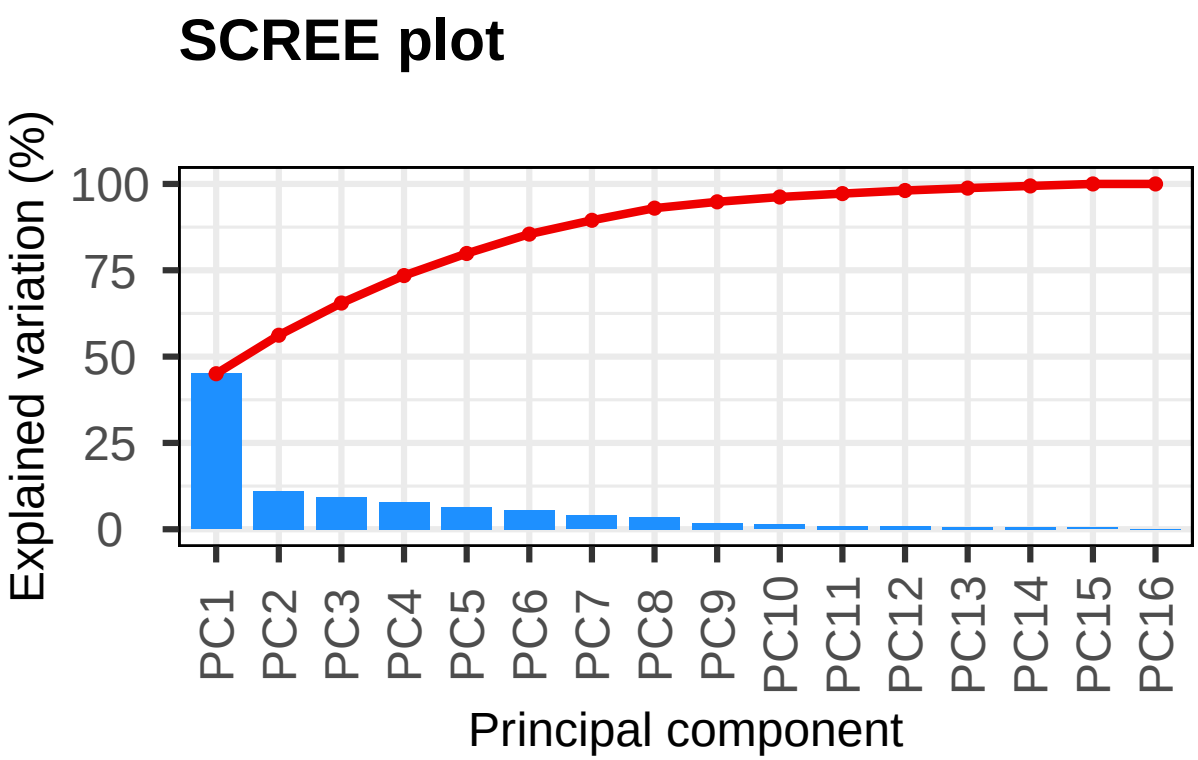
```
pcaData <- plotPCA(rld, intgroup=c("condition", "sample"), returnData=TRUE)
percentVar <- round(100 * attr(pcaData, "percentVar"))
ggplot(pcaData, aes(PC1, PC2, color=sample, shape=condition)) +
  geom_point(size=3) +
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance")) +
  coord_fixed()
```



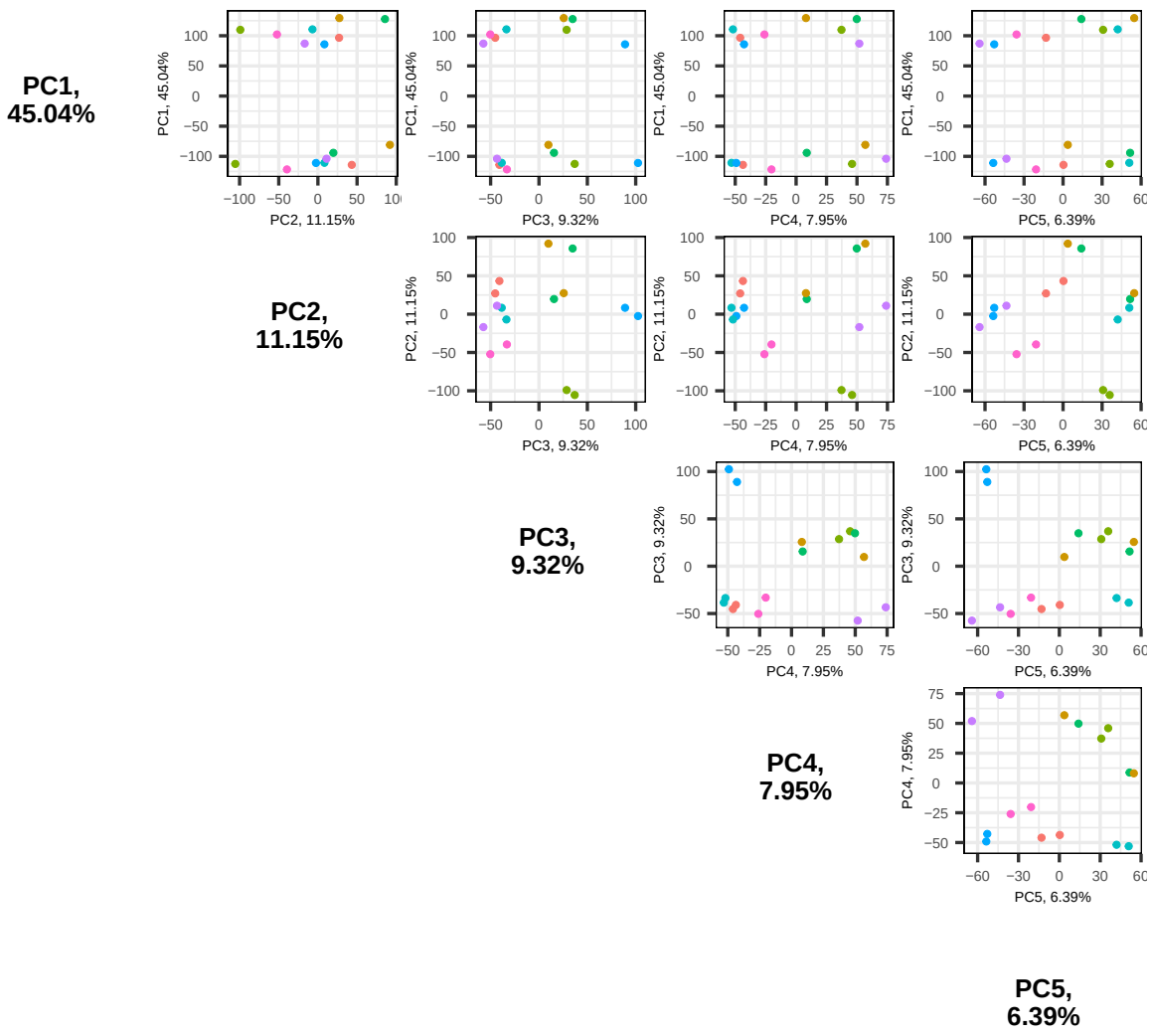
Okay, it's clear that PC1 is sequencing type.

For thoroughness, I'm going to check and see if sequencing is represented in PCs 3-5. I'm going to use the PCAtools tutorial (https://bioconductor.org/packages/release/bioc/vignettes/PCAtools/inst/doc/PCAtools.html) for this.

```
p <- pca(assay(rld), metadata = colData(rld), removeVar = 0.1)
screeplot(p, axisLabSize = 18, titleLabSize = 22)
```




```
pairsplot(p, colby = "sample", pointSize=2)
```

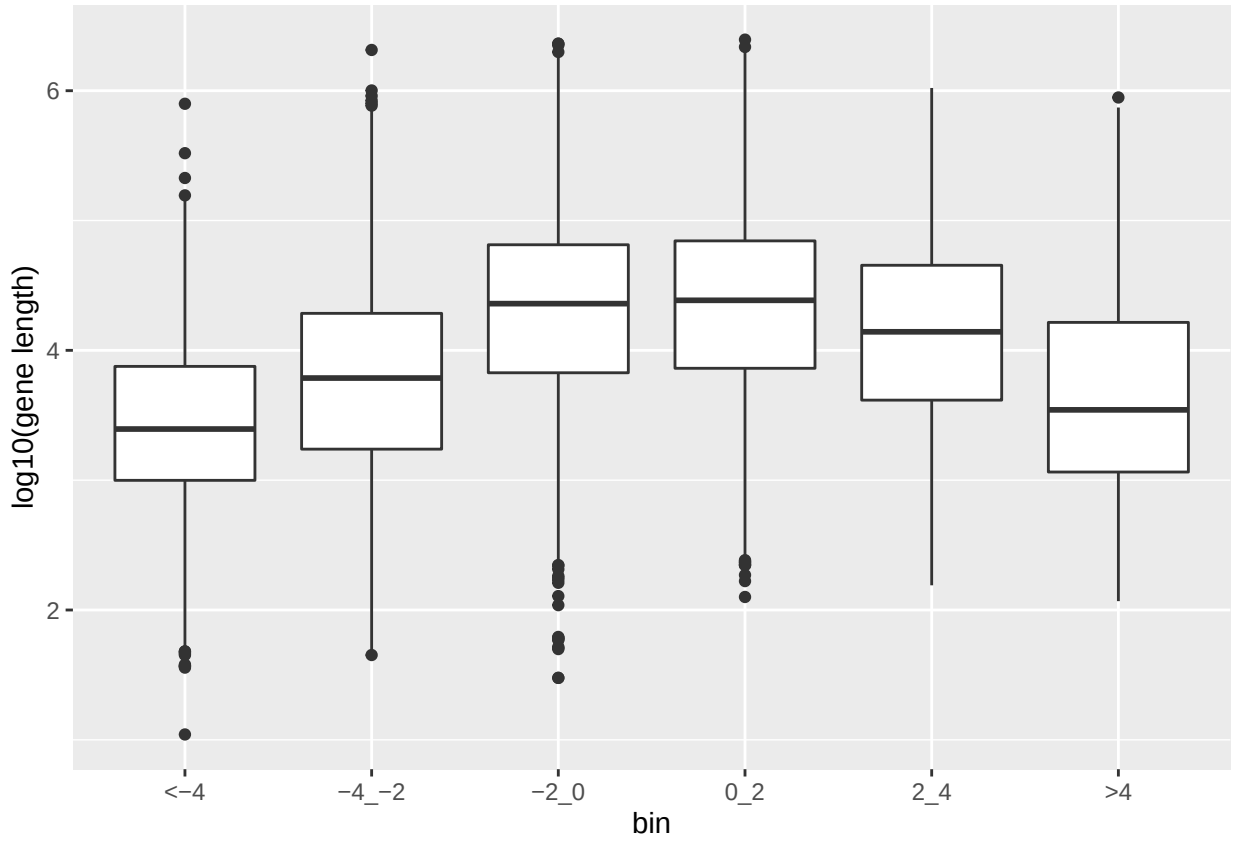


Okay, PCs 2-5 seem to be roughly based on sample, but the overwhelming distinction is sequencing type.

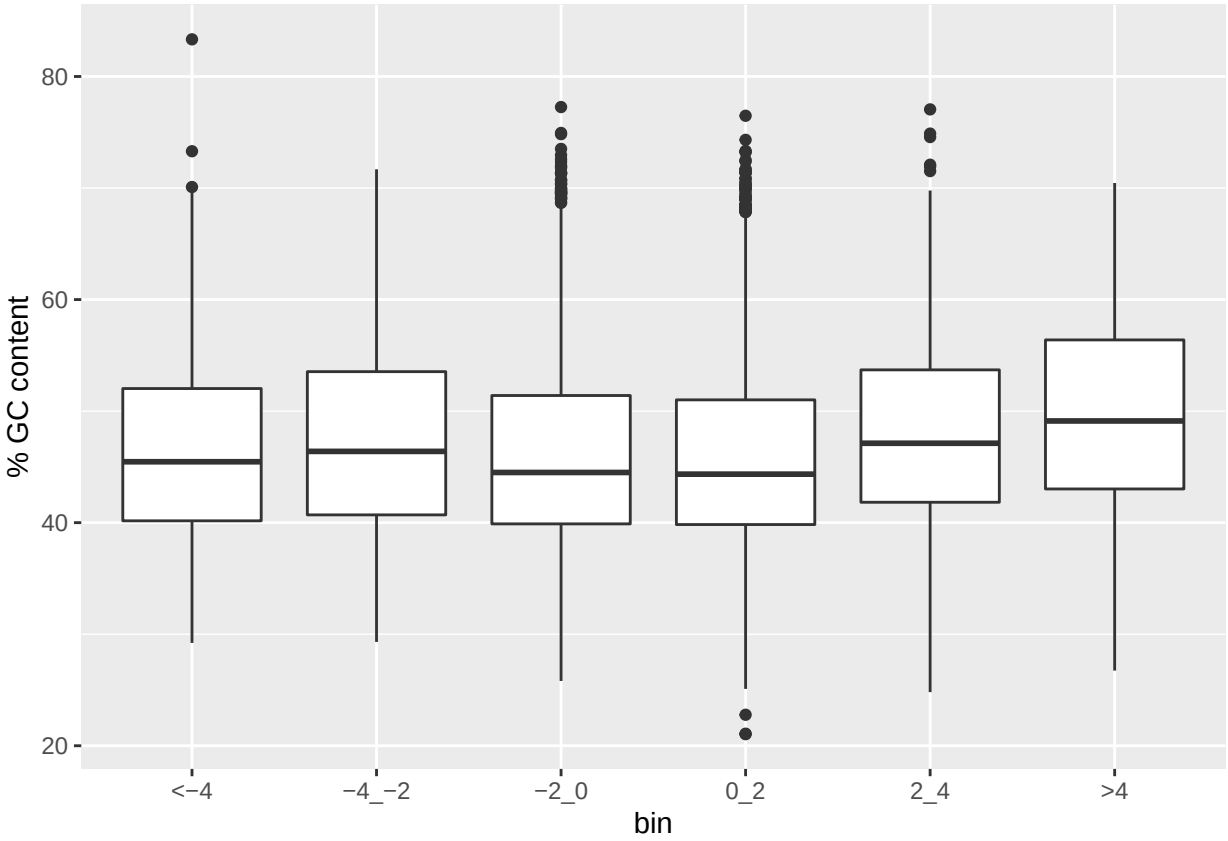
Checking gene length and GC content

```
resLFC$ensembl_gene_id <- rownames(resLFC)
res_df <- as.data.frame(resLFC)
res_df <- left_join(res_df, gene_coords)

res_df$bin <- "<-4"
res_df[res_df$log2FoldChange>=-4,]$bin <- "-4_-2"
res_df[res_df$log2FoldChange>=-2,]$bin <- "-2_0"
res_df[res_df$log2FoldChange>=0,]$bin <- "0_2"
```

```
res_df[res_df$log2FoldChange>=2,]$bin <- "2_4"
res_df[res_df$log2FoldChange>=4,]$bin <- ">4"
res_df$bin <- factor(res_df$bin, levels = c("<-4","-4_-2","-2_0","0_2","2_4",">4"))

ggplot(res_df, mapping = aes(x=bin, y=log10(size))) + geom_boxplot() + ylab("log10(gene length)")
```




```
ggplot(res_df, mapping = aes(x=bin, y=percentage_gene_gc_content)) + geom_boxplot() + ylab("% GC conten
```



Conclusions

The sequencing type seems to have such a big effect that I'm not sure we can really use pseudobulk as a control for differential expression. This is disappointing, but it may actually be a useful result to remind about the perils of using single-cell data for deconvolution. It's also a little surprising that there is significant differential expression in both directions, so this isn't just a factor of technical dropouts.

```
# Save data
deseq_path <- paste(local_data_path, "deseq2_output", sep = "/")
saveRDS(dds, file = paste(deseq_path, "polyA_vs_pseudo_data.rds", sep = "/"))

# Save results files
saveRDS(res, file = paste(deseq_path, "polyA_vs_pseudo_FDR_0.1.rds", sep = "/"))
saveRDS(res05, file = paste(deseq_path, "polyA_vs_pseudo_FDR_0.05.rds", sep = "/"))
```