TK_61: Rerun of RNAseq and RIBOseq datasets from samples induced with 2-5A/polyI:C

Hernan Lorenzi

7/17/2023

PI: Nick Guydosh
Point of Contact: Agnes Karasik
Contact email: agnes.karasik@nih.gov

Summary: The goal of this project is to recalculate Log2FC for RNAseq and RIBOseq datasets from samples induced with 2-5A/polyI:C, using read-count tables based on exon or CDS features.

Results

Link to filtered differential expression tables

Link to unfiltered differential expression tables

File	Contrast	Feature	Data source
DE_wtIC_vs_wtNone_cds.txt	WT - pIC vs Control	CDS	RNAseq
DE_koIC_vs_koNone_cds.txt	KO - pIC vs Control	CDS	RNAseq
DE_wt25A_vs_wtNone_cds.txt	WT - 2-5A vs Control	CDS	RNAseq
DE_ko25A_vs_koNone_cds.txt	KO - 2-5A vs Control	CDS	RNAseq
DE_wtIC_vs_wtNone_exon.txt	WT - pIC vs Control	Exon	RNAseq
DE_koIC_vs_koNone_exon.txt	KO - pIC vs Control	Exon	RNAseq
DE_wt25A_vs_wtNone_exon.txt	WT - 2-5A vs Control	Exon	RNAseq
DE_ko25A_vs_koNone_exon.txt	KO - 2-5A vs Control	Exon	RNAseq

DE_wtIC_vs_wtNone_RS.txt	WT - pIC vs Control	CDS	RIBOseq
DE_koIC_vs_koNone_RS.txt	KO - pIC vs Control	CDS	RIBOseq
DE_wt25A_vs_wtNone_RS.txt	WT - 2-5A vs Control	CDS	RIBOseq
DE_ko25A_vs_koNone_RS.txt	KO - 2-5A vs Control	CDS	RIBOseq

Link to exploratory plots

File	Contrast	Feature	Data source
barplot_read_counts_cds.pdf	Read counts per sample	CDS	RNAseq
pca_dataset_1_Induc_gt_cds.pdf	PCA colored by genotype	CDS	RNAseq
pca_dataset_1_Induc_read_counts_cds.pdf	PCA colored by sequencing depth	CDS	RNAseq
barplot_read_counts_exon.pdf	Read counts per sample	Exon	RNAseq
pca_dataset_1_Induc_gt_exon.pdf	PCA colored by genotype	Exon	RNAseq
pca_dataset_1_Induc_read_counts_exon.pdf	PCA colored by sequencing depth	Exon	RNAseq
barplot_read_counts_RS.pdf	Read counts per sample	CDS	RIBOseq
pca_dataset_1_Induc_gt_RS.pdf	PCA colored by genotype	CDS	RIBOseq
pca_dataset_1_Induc_read_counts_RS.pdf	PCA colored by sequencing depth	CDS	RIBOseq

Link to github repository

R code

Load libraries

```r
suppressMessages(library("org.Hs.eg.db"))
suppressMessages(library("pheatmap"))
#suppressMessages(library("EnhancedVolcano"))
suppressMessages(library("ggplot2"))
suppressMessages(library("ggpubr"))
suppressMessages(library("DESeq2"))
suppressMessages(library("stringr"))
suppressMessages(library("biomaRt"))
#suppressMessages(library("tidyverse"))
suppressMessages(library("pcaExplorer"))
#suppressMessages(library("clusterProfiler"))
suppressMessages(library("ggsci"))
suppressMessages(library("viridis"))
#suppressMessages(library("ggrepel"))
suppressMessages(library("RColorBrewer"))
#suppressMessages(library("msigdbr"))
suppressMessages(library("cowplot"))
#suppressMessages(library("enrichplot"))
#suppressMessages(library("ggupset"))
#suppressMessages(library("ggraph"))
```

Upload required functions

```r
# Load auxyliary functions
source(file = "./01_aux_rnaseq_functions.R")

# Load enrichment functions
source(file = "./02_Gene_enrichment_functions.R")
```

CDS counts

Load RNAseq CDS data

```
all <- read.delim2("./data/cdsrna_round.csv", sep = ",", header = TRUE, row.names

# Keep table with Ensemble IDs and gene Symbols
gene_symbols <- replace_gene_acc_by_symbol_ids(rownames(all))
ensembl_to_symbol <- as.data.frame(cbind("Ensembl_ID" = rownames(all), "gene_name"

# Load metadata
metadata <- read.delim2("./data/Metadata.txt", sep = "\t", row.names = 1, header =

# keep only samples that are present in all
metadata <- metadata[colnames(all),]

# Add total read counts and sample id columns to metadata
metadata <- cbind(metadata, Read_counts =colSums(all), Sample_id = rownames(metada

# Agnes wanted to keep all genes in the analysis, including all 0 genes
#Remove all zero rows
# all <- remove_all_zero_rows(all, min_total_count = 0)
```

Analysis of expression data using DESeq2 - CDS

```
# Convert metadata to factors
for (variable in c("Sequencing_pool", "Read_length","Machine", "Genotype","Group",
  metadata[,variable] <- as.factor(metadata[,variable])
}

# Subset metadata and count tables by Data
meta_one_cds <- subset(metadata, metadata$Dataset == "one")
all_one_cds <- all[, rownames(meta_one_cds)]
```

I created a new column in metadata (Group_gt_ind) that concatenates the info from Genotype and Inducer columns so coefficients include genotype info.

```
dir.create(path = "./Plots", showWarnings = F)
dir.create(path = "./DE", showWarnings = F)

meta_one_cds$Group_gt_ind <- factor(paste0(meta_one_cds$Genotype, meta_one_cds$Inc
```

I also added a new column (Read_depth) to tag samples with High or Low sequencing depth so this factor can be controlled for in the design formula.

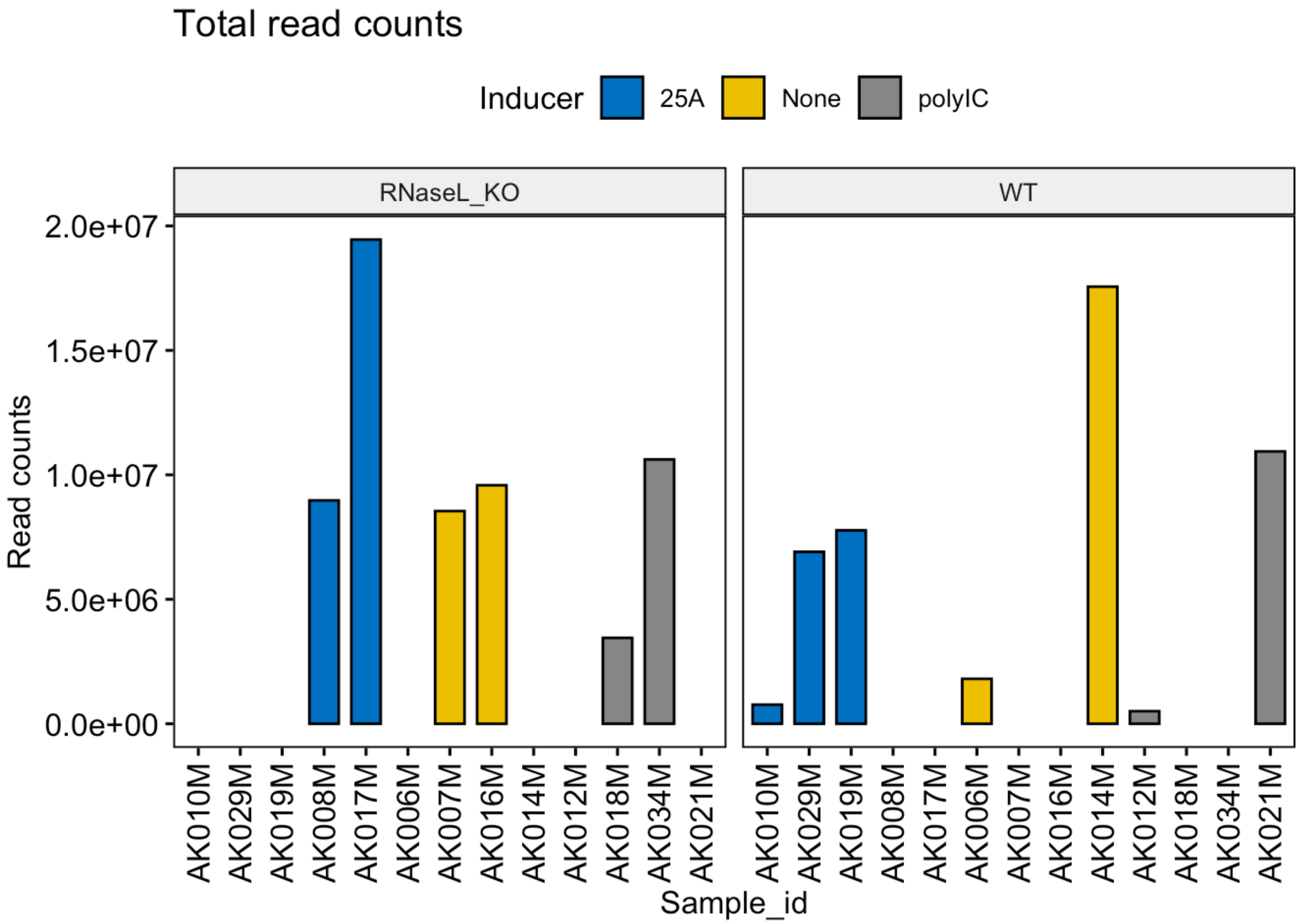
The design formula use in DESeq2 is the following:

```
design = ~ Read_depth + Group_gt_ind
```

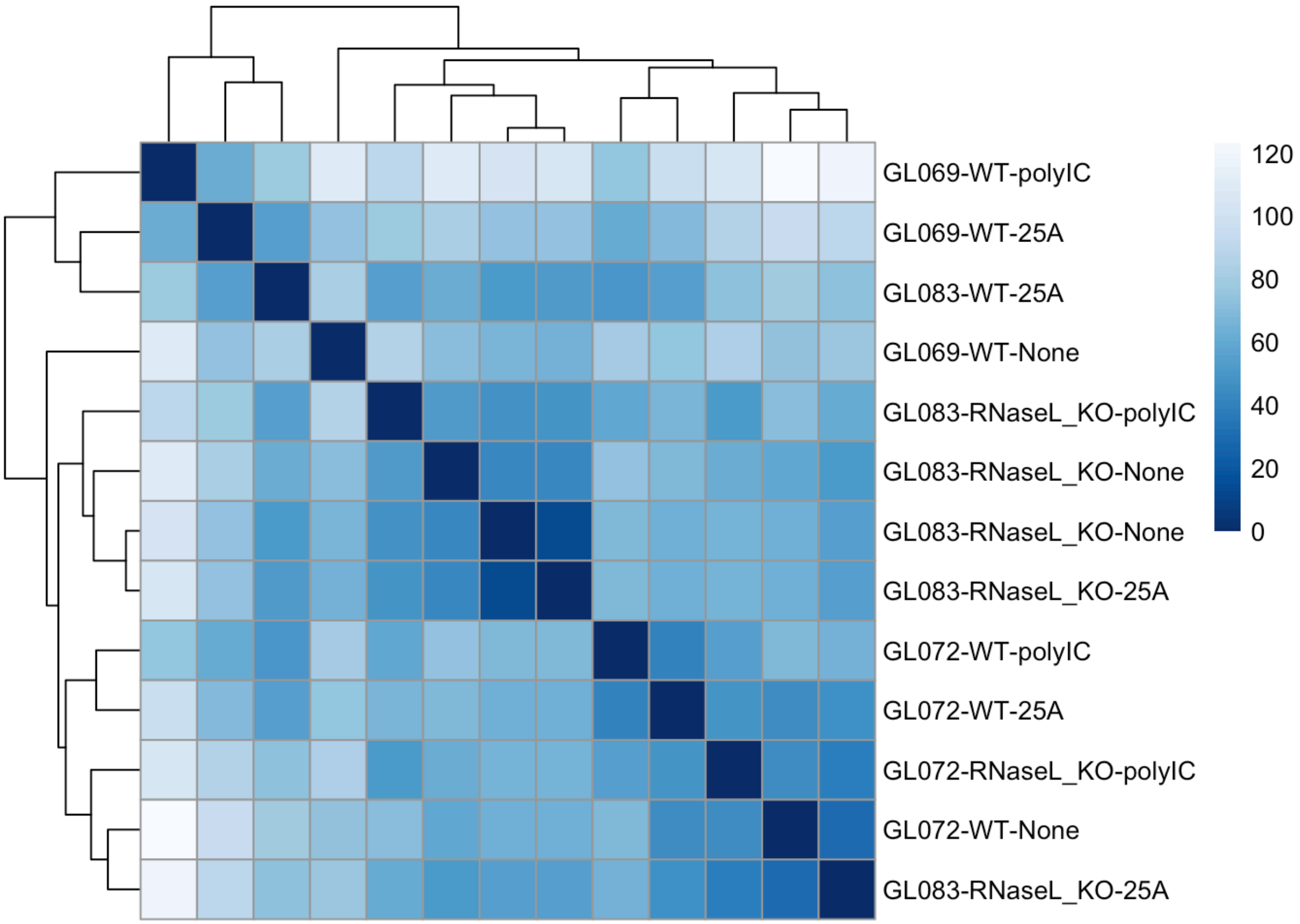
```
# add new factors (Group_gt_ind and Read_depth (high > 3M reads / Low < 3M reads)
meta_one_cds$Read_depth <- 'High'
meta_one_cds[meta_one_cds$Read_counts < 3e6,]$Read_depth <- 'Low'
meta_one_cds$Read_depth <- as.factor(meta_one_cds$Read_depth)

# Adding read_depth in design to control for read_depth
dds.one.cds <- DESeqDataSetFromMatrix(countData = all_one_cds,
                              colData = meta_one_cds,
                              design = ~ Read_depth + Group_gt_ind)
```

Exploratory analysis with DESeq object- CDS

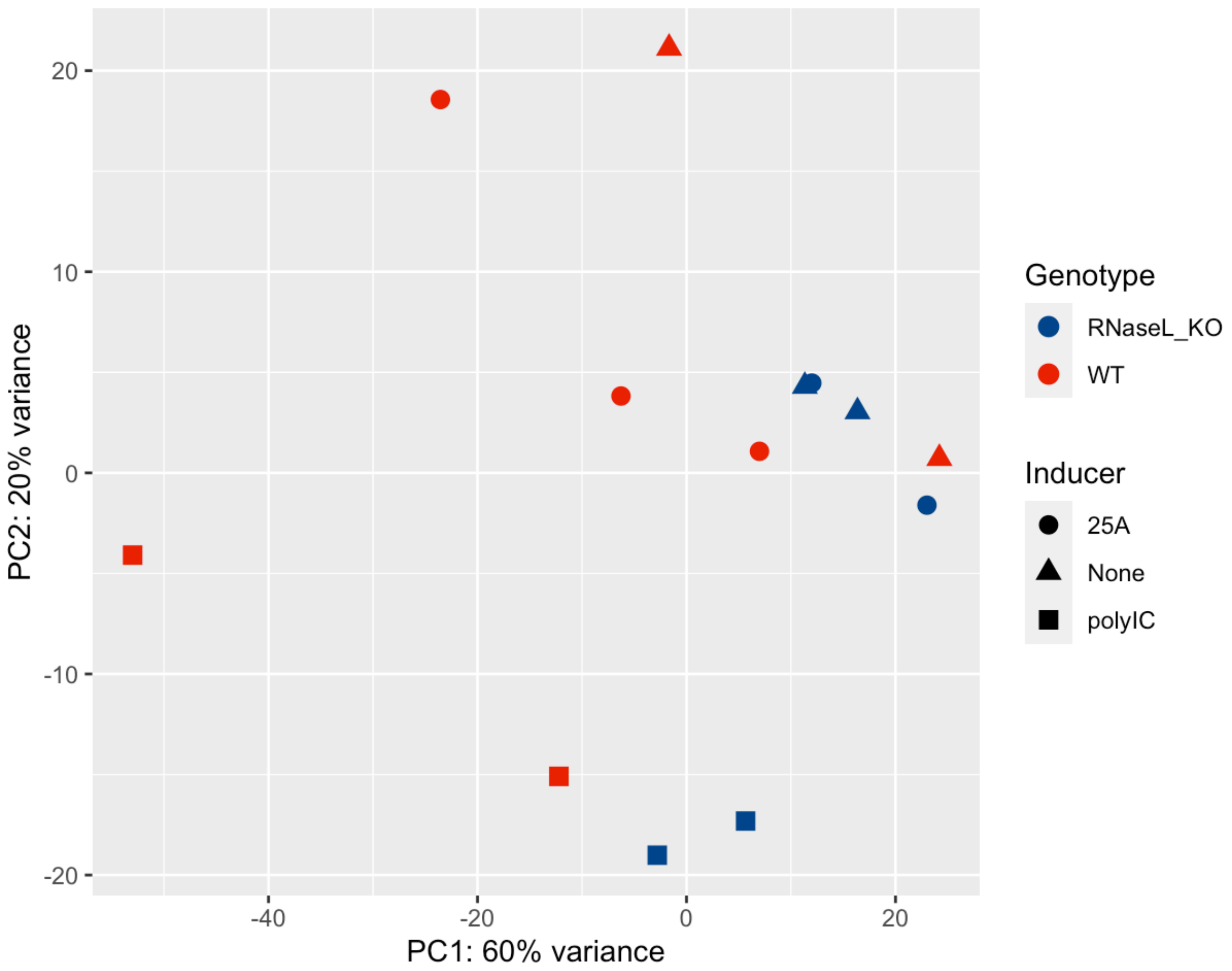
```
# Plot total reads per sample using barchar
p <- ggbarplot(data = meta_one_cds,
          x = "Sample_id",
          y = "Read_counts",
          x.text.angle = 90,
          fill = "Inducer",
          title = "Total read counts",
          ylab = "Read counts",
          sort.by.groups = TRUE,
          palette = "jco",
          sort.val = "asc",
          facet.by = "Genotype")
ggsave("Plots/barplot_read_counts_cds.pdf", plot = p)
p
```



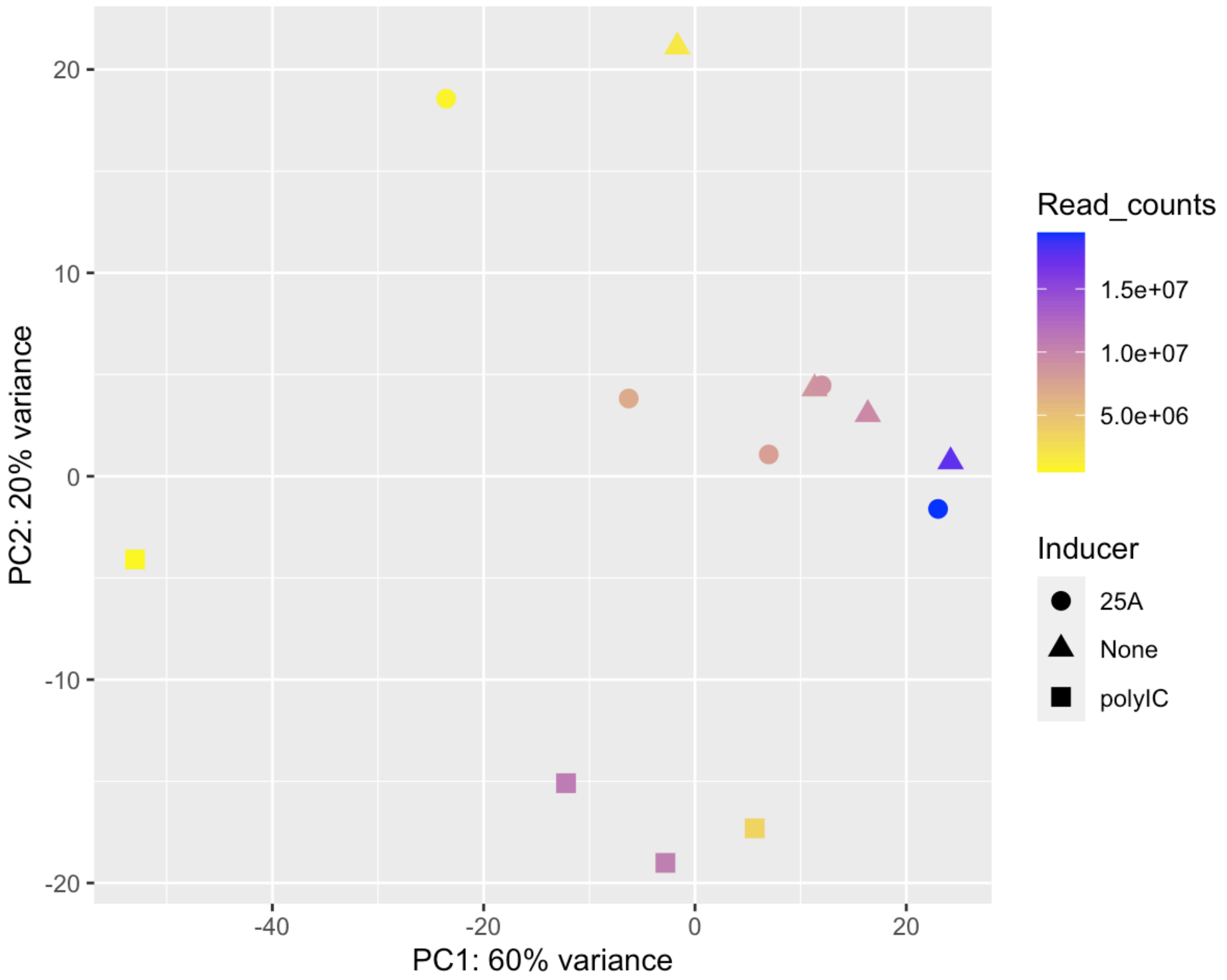
```
# Normalize counts
rlog.one <- rlog(dds.one.cds, blind=FALSE)

# Keep genes with at least 0 reads total across samples
keep <- rowSums(counts(dds.one.cds)) >= 0 # Agnes wanted to keep all genes for the
dds.one.cds <- dds.one.cds[keep,]

# Calculate distances between samples
sampleDists <- dist(t(assay(rlog.one)))

# Plot inter-sample distances
old.par <- par(no.readonly=T)

sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(rlog.one$Sequencing_pool, rlog.one$Genotype,
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors)
```



```
# PCA
my_top_genes = 540

pcaData <- plotPCA(rlog.one, intgroup=c("Genotype", "Inducer"), returnData=TRUE, r
percentVar <- round(100 * attr(pcaData, "percentVar"))
y.coords = c(min(pcaData$PC1, pcaData$PC2), max(pcaData$PC1, pcaData$PC2))
x.coords = y.coords
p1 <- ggplot(pcaData, aes(PC1, PC2, color=Genotype, shape=Inducer)) +
  geom_point(size=3) + scale_color_lancet() +
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance")) +
  coord_fixed(ratio = (max(pcaData$PC1)-min(pcaData$PC1))/(max(pcaData$PC2)-min(pc

ggsave("Plots/pca_dataset_1_Induc_gt_cds.pdf", plot = p1)
p1
```



```
pcaData <- plotPCA(rlog.one, intgroup=c("Read_counts", "Inducer"), returnData=TRUE
percentVar <- round(100 * attr(pcaData, "percentVar"))
p2 <- ggplot(pcaData, aes(PC1, PC2, color=Read_counts, shape=Inducer)) +
  geom_point(size=3) +
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance")) +
  coord_fixed(ratio = (max(pcaData$PC1)-min(pcaData$PC1))/(max(pcaData$PC2)-min(pc

ggsave("Plots/pca_dataset_1_Induc_read_counts_cds.pdf", plot = p2)
p2
```



PCA plots indicate that samples separated on PC1 mostly by sequencing depth, within treatments and genotypes. This implies that sequencing depth has to be controlled for by including this factor in the design formula.

Filtering out poorly-expressed genes (less than 20 reads across all samples) - CDS

```
# Keep genes with at least 10 reads total across samples
keep <- rowSums(counts(dds.one.cds)) >= 0 # Agnes wanted to keep all genes
dds.one.cds <- dds.one.cds[keep,]
```

Spliting DESeq object based on genotype - CDS

```
dds.one.cds.wt <- dds.one.cds[ , dds.one.cds$Genotype == "WT"]
dds.one.cds.wt$Genotype <- droplevels( dds.one.cds.wt$Genotype)
dds.one.cds.wt$Group_gt_ind <- droplevels( dds.one.cds.wt$Group_gt_ind)
dds.one.cds.wt$Group <- droplevels( dds.one.cds.wt$Group)

dds.one.cds.ko <- dds.one.cds[ , dds.one.cds$Genotype == "RNaseL_KO"]
dds.one.cds.ko$Genotype <- droplevels( dds.one.cds.ko$Genotype)
dds.one.cds.ko$Group_gt_ind <- droplevels( dds.one.cds.ko$Group_gt_ind)
dds.one.cds.ko$Group <- droplevels( dds.one.cds.ko$Group)
```

Calculate differential expression for WT - CDS

```
# Calculate DE for WT samples
dds.one.cds.wt$Group_gt_ind <- relevel(dds.one.cds.wt$Group_gt_ind, "WTNone")
dds.one.cds.wt <- DESeq(dds.one.cds.wt)
resultsNames(dds.one.cds.wt)
```

```
## [1] "Intercept"                       "Read_depth_Low_vs_High"
## [3] "Group_gt_ind_WT25A_vs_WTNone"    "Group_gt_ind_WTpolyIC_vs_WTNone"
```

```
# Using lfcShrink instead of results to reduce high Log2FC bias of genes with low
res_wtIC_vs_wtNone <- lfcShrink(dds.one.cds.wt, coef = "Group_gt_ind_WTpolyIC_vs_W
res_wt25A_vs_wtNone <- lfcShrink(dds.one.cds.wt, coef = "Group_gt_ind_WT25A_vs_WTN

summary(res_wtIC_vs_wtNone, alpha = 0.05)
```

```
##
## out of 17368 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 1208, 7%
## LFC < 0 (down)     : 644, 3.7%
## outliers [1]       : 0, 0%
## low counts [2]     : 4973, 29%
## (mean count < 8)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
summary(res_wt25A_vs_wtNone, alpha = 0.05)
```

```
##
## out of 17368 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 730, 4.2%
## LFC < 0 (down)     : 225, 1.3%
## outliers [1]       : 0, 0%
## low counts [2]     : 7294, 42%
## (mean count < 35)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Calculate DE for KO samples - CDS

```
dds.one.cds.ko$Group_gt_ind <- relevel(dds.one.cds.ko$Group_gt_ind, "RNaseL_KONone
# Changing design formula given that there is no KO sample with Low read counts, c
design(dds.one.cds.ko) <- ~Group_gt_ind
dds.one.cds.ko <- DESeq(dds.one.cds.ko)
resultsNames(dds.one.cds.ko)
```

```
## [1] "Intercept"
## [2] "Group_gt_ind_RNaseL_KO25A_vs_RNaseL_KONone"
## [3] "Group_gt_ind_RNaseL_KOpolyIC_vs_RNaseL_KONone"
```

```
res_koIC_vs_koNone <- lfcShrink(dds.one.cds.ko, coef = "Group_gt_ind_RNaseL_KOpoly
res_ko25A_vs_koNone <- lfcShrink(dds.one.cds.ko, coef = "Group_gt_ind_RNaseL_KO25A

summary(res_koIC_vs_koNone, alpha = 0.05)
```

```
##
## out of 17051 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 142, 0.83%
## LFC < 0 (down)     : 3, 0.018%
## outliers [1]       : 0, 0%
## low counts [2]     : 1281, 7.5%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
summary(res_ko25A_vs_koNone, alpha = 0.05)
```

```
##
```

```
## out of 17051 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 0, 0%
## LFC < 0 (down)     : 0, 0%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Write DE tables to file - CDS

```
# Define function for processing and saving result tables
sort_and_write_res_table <- function(result_table, file_name){
  # Sort genes by (padj)
  result_table_sorted <- result_table[order(result_table$padj, decreasing = FALSE
  # Add gene symbols
  gene_list <- rownames(result_table_sorted)
  symbol_list <- ensembl_to_symbol$gene_name[match(gene_list, ensembl_to_symbol$E
  df <-as.data.frame(cbind(result_table_sorted, Gene_name = symbol_list))

  # Write sorted table to file
  write.table(df, file = paste0("./DE/",file_name,".txt"),
            sep = "\t", col.names=NA)
  return(result_table_sorted)
}

# Sort results by Log2FC
res_wtIC_vs_wtNone.logfc_sorted <- sort_and_write_res_table(res_wtIC_vs_wtNone, "
res_koIC_vs_koNone.logfc_sorted <- sort_and_write_res_table(res_koIC_vs_koNone, "
res_wt25A_vs_wtNone.logfc_sorted <- sort_and_write_res_table(res_wt25A_vs_wtNone,
res_ko25A_vs_koNone.logfc_sorted <- sort_and_write_res_table(res_ko25A_vs_koNone,

# Save sorted files as a list
DE_results = list()
DE_results[["wtIC_vs_wtNone_cds"]]  <- res_wtIC_vs_wtNone.logfc_sorted
DE_results[["koIC_vs_koNone_cds"]] <- res_koIC_vs_koNone.logfc_sorted
DE_results[["wt25A_vs_wtNone_cds"]] <- res_wt25A_vs_wtNone.logfc_sorted
DE_results[["ko25A_vs_koNone_cds"]] <- res_ko25A_vs_koNone.logfc_sorted
```

Exonic counts

Load data - exons

```
all <- read.delim2("./data/read_counts_exonic_RNaseL.csv", sep = ",", header = TRU

# Make sure read counts are numeric and rounded to 0 decimals
all.tmp <- as.data.frame(lapply(all, function(x){ round(as.numeric(x), digits = 0)
rownames(all.tmp) <- rownames(all)
all <- all.tmp

# Keep table with Ensemble IDs and gene Symbols
gene_symbols <- replace_gene_acc_by_symbol_ids(rownames(all))
ensembl_to_symbol <- as.data.frame(cbind("Ensembl_ID" = rownames(all), "gene_name"

# Load metadata
metadata <- read.delim2("./data/Metadata.txt", sep = "\t", row.names = 1, header =

# keep only samples that are present in all
metadata <- metadata[colnames(all),]

# Add total read counts and sample id columns to metadata
metadata <- cbind(metadata, Read_counts =colSums(all), Sample_id = rownames(metada

# Agnes wanted to keep all genes in the analysis, including all 0 genes
#Remove all zero rows
#all <- remove_all_zero_rows(all, min_total_count = 0)
```

Analysis of expression data using DESeq2 - exons

```
# Convert metadata to factors
for (variable in c("Sequencing_pool", "Read_length","Machine", "Genotype","Group",
  metadata[,variable] <- as.factor(metadata[,variable])
}

# Subset metadata and count tables by Data
meta_one_exon <- subset(metadata, metadata$Dataset == "one")
all_one_exon <- all[, rownames(meta_one_exon)]
```

I created a new column in metadata (Group_gt_ind) that concatenates the info from Genotype and

Inducer columns so coefficients include genotype info.

I also added a new column (Read_depth) to tag samples with High or Low sequencing depth so this factor can be controlled for in the design formula.

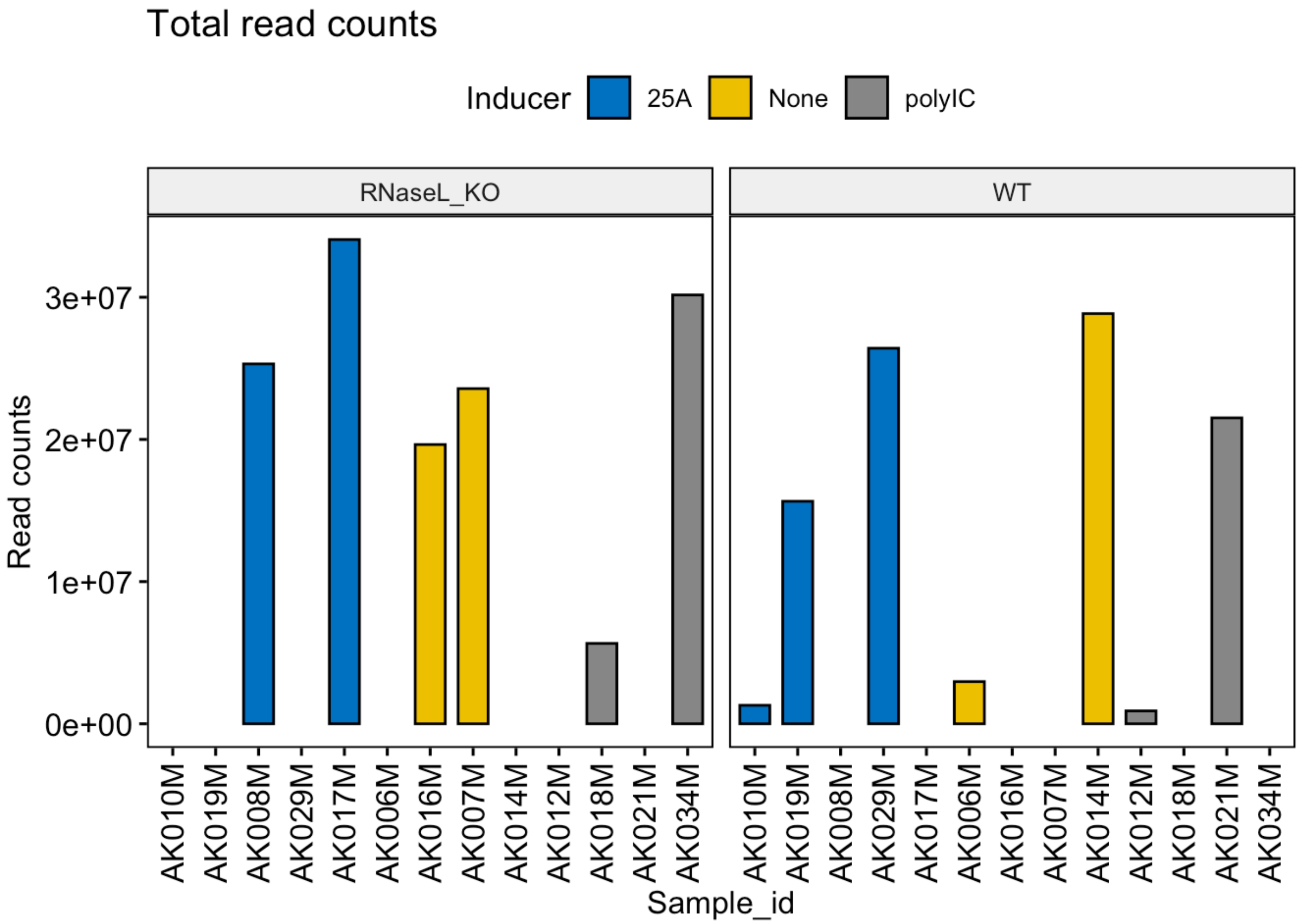
The design formula use in DESeq2 is the following:

```
design = ~ Read_depth + Group_gt_ind
```

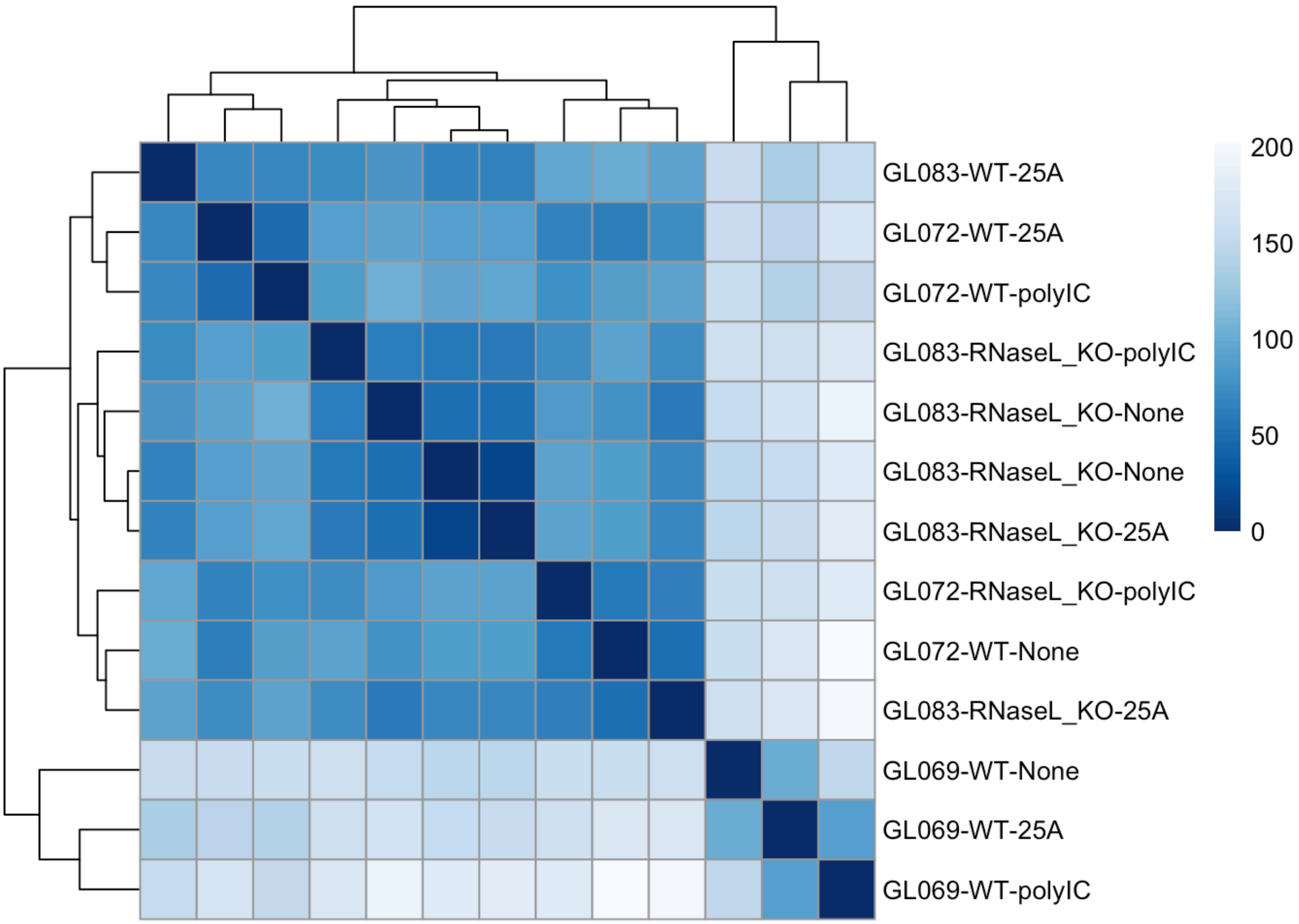
```
dir.create(path = "./Plots", showWarnings = F)
dir.create(path = "./DE", showWarnings = F)

meta_one_exon$Group_gt_ind <- factor(paste0(meta_one_exon$Genotype, meta_one_exon$

# add new factors (Group_gt_ind and Read_depth (high > 10M reads / Low < 10M reads
meta_one_exon$Read_depth <- 'High'
meta_one_exon[meta_one_exon$Read_counts < 10e6,]$Read_depth <- 'Low'
meta_one_exon$Read_depth <- as.factor(meta_one_exon$Read_depth)

# Adding read_depth in design to control for read_depth
dds.one.exon <- DESeqDataSetFromMatrix(countData = all_one_exon,
                              colData = meta_one_exon,
                              design = ~ Read_depth + Group_gt_ind)
```

Exploratory analysis with DESeq object- exons

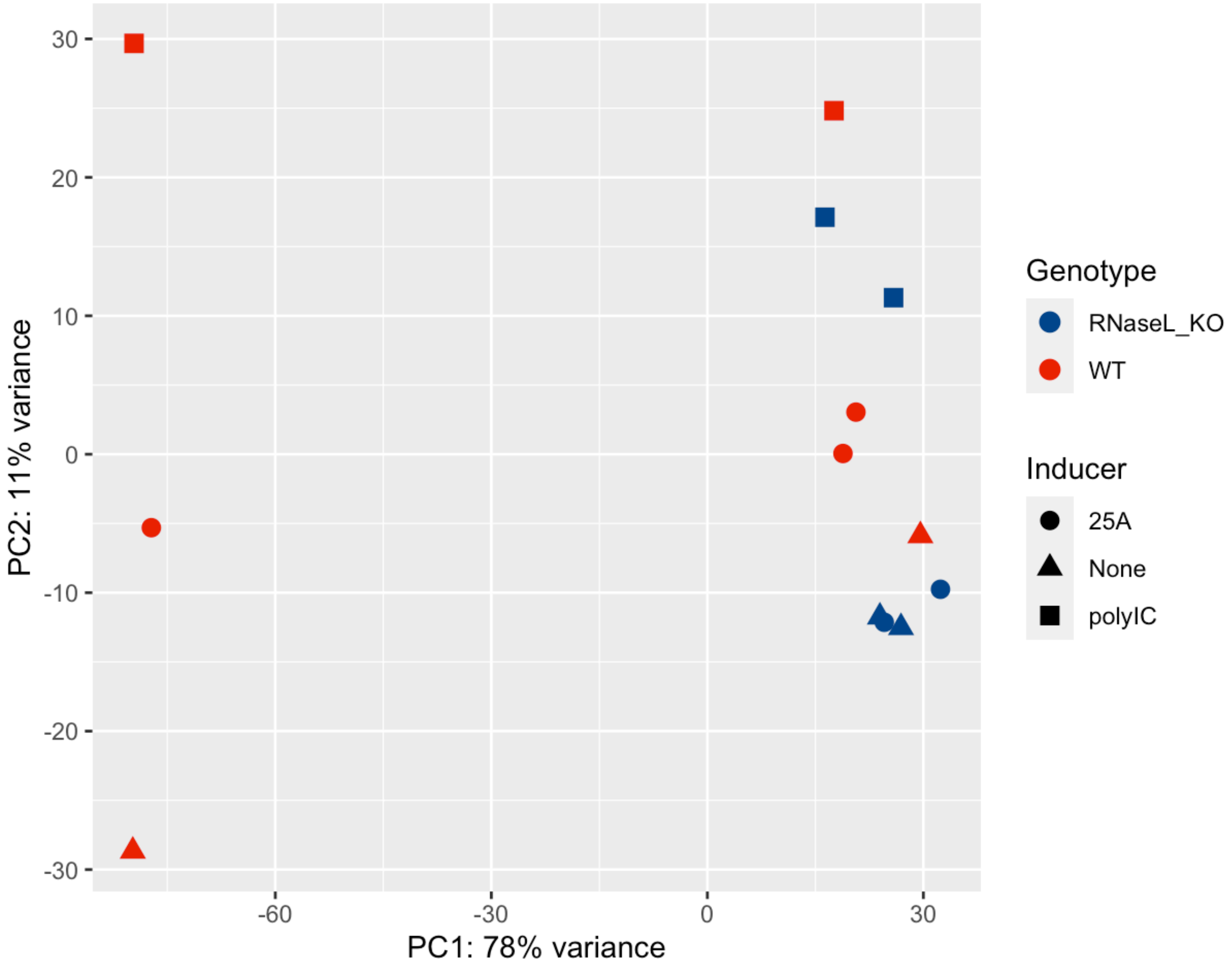
```
# Plot total reads per sample using barchar
p <- ggbarplot(data = meta_one_exon,
          x = "Sample_id",
          y = "Read_counts",
          x.text.angle = 90,
          fill = "Inducer",
          title = "Total read counts",
          ylab = "Read counts",
          sort.by.groups = TRUE,
          palette = "jco",
          sort.val = "asc",
          facet.by = "Genotype")
ggsave("Plots/barplot_read_counts_exon.pdf", plot = p)
p
```



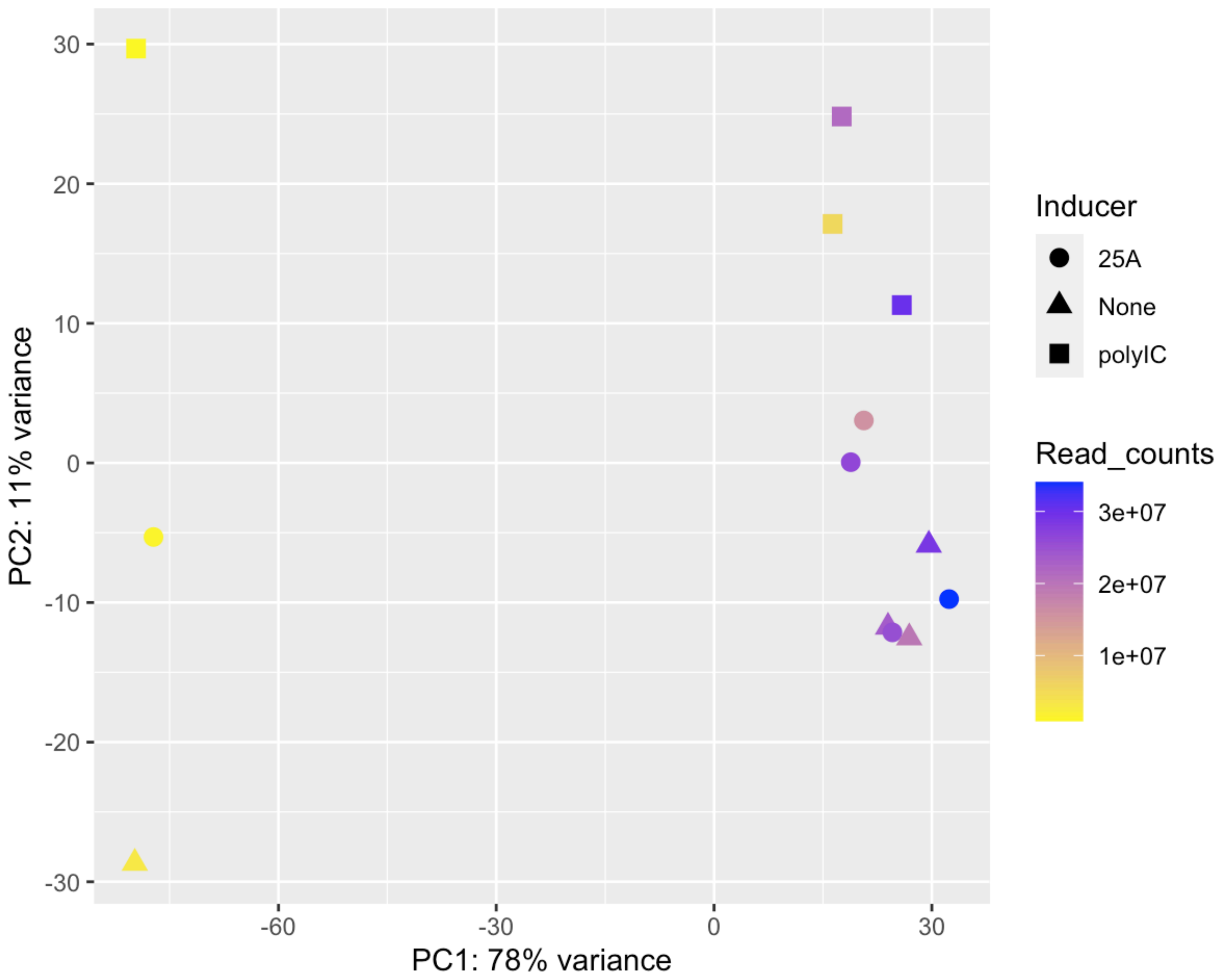
```
# Normalize counts
rlog.one <- rlog(dds.one.exon, blind=FALSE)

# Keep genes with at least 20 reads total across samples
keep <- rowSums(counts(dds.one.exon)) >= 0 # Agnes wanted to keep all genes for t
dds.one.exon <- dds.one.exon[keep,]

# Calculate distances between samples
sampleDists <- dist(t(assay(rlog.one)))

# Plot inter-sample distances
old.par <- par(no.readonly=T)

sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(rlog.one$Sequencing_pool, rlog.one$Genotype,
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors)
```



```
# PCA
my_top_genes = 540

pcaData <- plotPCA(rlog.one, intgroup=c("Genotype", "Inducer"), returnData=TRUE, n
percentVar <- round(100 * attr(pcaData, "percentVar"))
y.coords = c(min(pcaData$PC1, pcaData$PC2), max(pcaData$PC1, pcaData$PC2))
x.coords = y.coords
p1 <- ggplot(pcaData, aes(PC1, PC2, color=Genotype, shape=Inducer)) +
  geom_point(size=3) + scale_color_lancet() +
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance")) +
  coord_fixed(ratio = (max(pcaData$PC1)-min(pcaData$PC1))/(max(pcaData$PC2)-min(pc

ggsave("Plots/pca_dataset_1_Induc_gt_exon.pdf", plot = p1)
p1
```



```
pcaData <- plotPCA(rlog.one, intgroup=c("Read_counts", "Inducer"), returnData=TRUE
percentVar <- round(100 * attr(pcaData, "percentVar"))
p2 <- ggplot(pcaData, aes(PC1, PC2, color=Read_counts, shape=Inducer)) +
  geom_point(size=3) +
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance")) +
  coord_fixed(ratio = (max(pcaData$PC1)-min(pcaData$PC1))/(max(pcaData$PC2)-min(pc

ggsave("Plots/pca_dataset_1_Induc_read_counts_exon.pdf", plot = p2)
p2
```



PCA plots indicate that samples separated on PC1 mostly by sequencing depth, within treatments and genotypes. This implies that sequencing depth has to be controlled for by including this factor in the design formula.

Filtering out poorly-expressed genes (less than 20 reads across all samples) - exons

```
# Keep genes with at least 10 reads total across samples
keep <- rowSums(counts(dds.one.exon)) >= 0 # Agnes wanted to keep all genes
dds.one.exon <- dds.one.exon[keep,]
```

Spliting DESeq object based on genotype - exons

```
dds.one.exon.wt <- dds.one.exon[ , dds.one.exon$Genotype == "WT"]
dds.one.exon.wt$Genotype <- droplevels( dds.one.exon.wt$Genotype)
dds.one.exon.wt$Group_gt_ind <- droplevels( dds.one.exon.wt$Group_gt_ind)
dds.one.exon.wt$Group <- droplevels( dds.one.exon.wt$Group)

dds.one.exon.ko <- dds.one.exon[ , dds.one.exon$Genotype == "RNaseL_KO"]
dds.one.exon.ko$Genotype <- droplevels( dds.one.exon.ko$Genotype)
dds.one.exon.ko$Group_gt_ind <- droplevels( dds.one.exon.ko$Group_gt_ind)
dds.one.exon.ko$Group <- droplevels( dds.one.exon.ko$Group)
```

Calculate differential expression for WT - exons

```
# Calculate DE for WT samples
dds.one.exon.wt$Group_gt_ind <- relevel(dds.one.exon.wt$Group_gt_ind, "WTNone")
dds.one.exon.wt <- DESeq(dds.one.exon.wt)
resultsNames(dds.one.exon.wt)
```

```
## [1] "Intercept"                          "Read_depth_Low_vs_High"
## [3] "Group_gt_ind_WT25A_vs_WTNone"       "Group_gt_ind_WTpolyIC_vs_WTNone"
```

```
# Using lfcShrink instead of results to reduce high Log2FC bias of genes with low
res_wtIC_vs_wtNone <- lfcShrink(dds.one.exon.wt, coef = "Group_gt_ind_WTpolyIC_vs_
res_wt25A_vs_wtNone <- lfcShrink(dds.one.exon.wt, coef = "Group_gt_ind_WT25A_vs_W

summary(res_wtIC_vs_wtNone, alpha = 0.05)
```

```
##
## out of 37152 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 1707, 4.6%
## LFC < 0 (down)     : 1720, 4.6%
## outliers [1]       : 0, 0%
## low counts [2]     : 19394, 52%
## (mean count < 7)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
summary(res_wt25A_vs_wtNone, alpha = 0.05)
```

```
##
## out of 37152 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 1051, 2.8%
## LFC < 0 (down)     : 882, 2.4%
## outliers [1]       : 0, 0%
## low counts [2]     : 22857, 62%
## (mean count < 20)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Calculate differential expression for RNaseL_KO - exons

```
dds.one.exon.ko$Group_gt_ind <- relevel(dds.one.exon.ko$Group_gt_ind, "RNaseL_KON
design(dds.one.exon.ko) <- ~Group_gt_ind # Changing design given that there is no
# Error: full model matrix is less than full rank
dds.one.exon.ko <- DESeq(dds.one.exon.ko)
resultsNames(dds.one.exon.ko)
```

```
## [1] "Intercept"
## [2] "Group_gt_ind_RNaseL_KO25A_vs_RNaseL_KONone"
## [3] "Group_gt_ind_RNaseL_KOpolyIC_vs_RNaseL_KONone"
```

```
res_koIC_vs_koNone <- lfcShrink(dds.one.exon.ko, coef = "Group_gt_ind_RNaseL_KOpo
res_ko25A_vs_koNone <- lfcShrink(dds.one.exon.ko, coef = "Group_gt_ind_RNaseL_KO25

summary(res_koIC_vs_koNone, alpha = 0.05)
```

```
##
## out of 36014 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 171, 0.47%
## LFC < 0 (down)     : 4, 0.011%
## outliers [1]       : 0, 0%
## low counts [2]     : 16066, 45%
## (mean count < 4)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
summary(res_ko25A_vs_koNone, alpha = 0.05)
```

```
##
## out of 36014 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 0, 0%
## LFC < 0 (down)     : 0, 0%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Write DE tables to file - exons

```
# Sort results by Log2FC
res_wtIC_vs_wtNone.logfc_sorted <- sort_and_write_res_table(res_wtIC_vs_wtNone, "
res_koIC_vs_koNone.logfc_sorted <- sort_and_write_res_table(res_koIC_vs_koNone, "
res_wt25A_vs_wtNone.logfc_sorted <- sort_and_write_res_table(res_wt25A_vs_wtNone,
res_ko25A_vs_koNone.logfc_sorted <- sort_and_write_res_table(res_ko25A_vs_koNone,

# Save sorted files as a list
DE_results[["wtIC_vs_wtNone_exon"]]  <- res_wtIC_vs_wtNone.logfc_sorted
DE_results[["koIC_vs_koNone_exon"]] <- res_koIC_vs_koNone.logfc_sorted
DE_results[["wt25A_vs_wtNone_exon"]] <- res_wt25A_vs_wtNone.logfc_sorted
DE_results[["ko25A_vs_koNone_exon"]] <- res_ko25A_vs_koNone.logfc_sorted
```

RIBOseq counts

Load data - RIBOseq

```
all <- read.csv("./data/cdsfoot_round.csv", row.names = 1)

# Keep table with Ensemble IDs and gene Symbols
gene_symbols <- replace_gene_acc_by_symbol_ids(rownames(all))
ensembl_to_symbol <- as.data.frame(cbind("Ensembl_ID" = rownames(all), "gene_name"

# Load metadata
metadata <- read.delim2("./data/Metadata_footprint.txt", sep = "\t", row.names = 
# Sort tables so metadata and read counts match order
metadata<-  metadata[match(colnames(all), rownames(metadata)), ]

# Add total read counts and sample id columns to metadata
metadata <- cbind(metadata, Read_counts =colSums(all), Sample_id = rownames(metada

# Agnes wanted to keep all genes in the analysis, including all 0 genes
#Remove all zero rows
#all <- remove_all_zero_rows(all, min_total_count = 0)
```

Analysis of expression data using DESeq2 - RIBOseq

```
# Convert metadata to factors
for (variable in c("Read_length","Sequencing_pool","Date_1st_submitted","Species",
  metadata[,variable] <- as.factor(metadata[,variable])
}

# Subset metadata and count tables by Data
meta_one_RS <- metadata
all_one_RS <- all
```

I created a new column in metadata (Group_gt_ind) that concatenates the info from Genotype and Inducer columns so coefficients include genotype info.

I also added a new column (Read_depth) to tag samples with High or Low sequencing depth so this factor can be controlled for in the design formula.

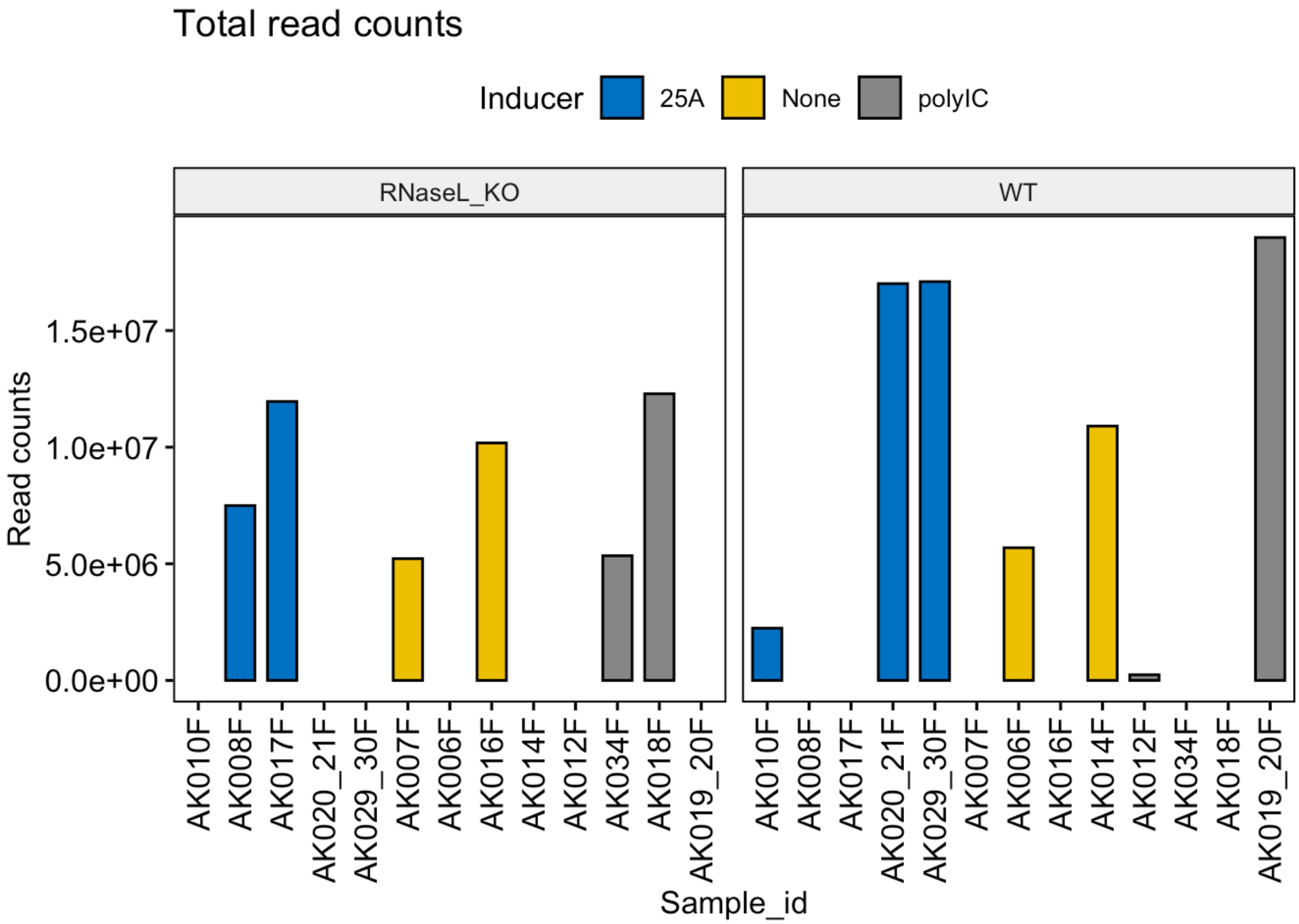
The design formula use in DESeq2 is the following:

```
design = ~ Read_depth + Group_gt_ind
```

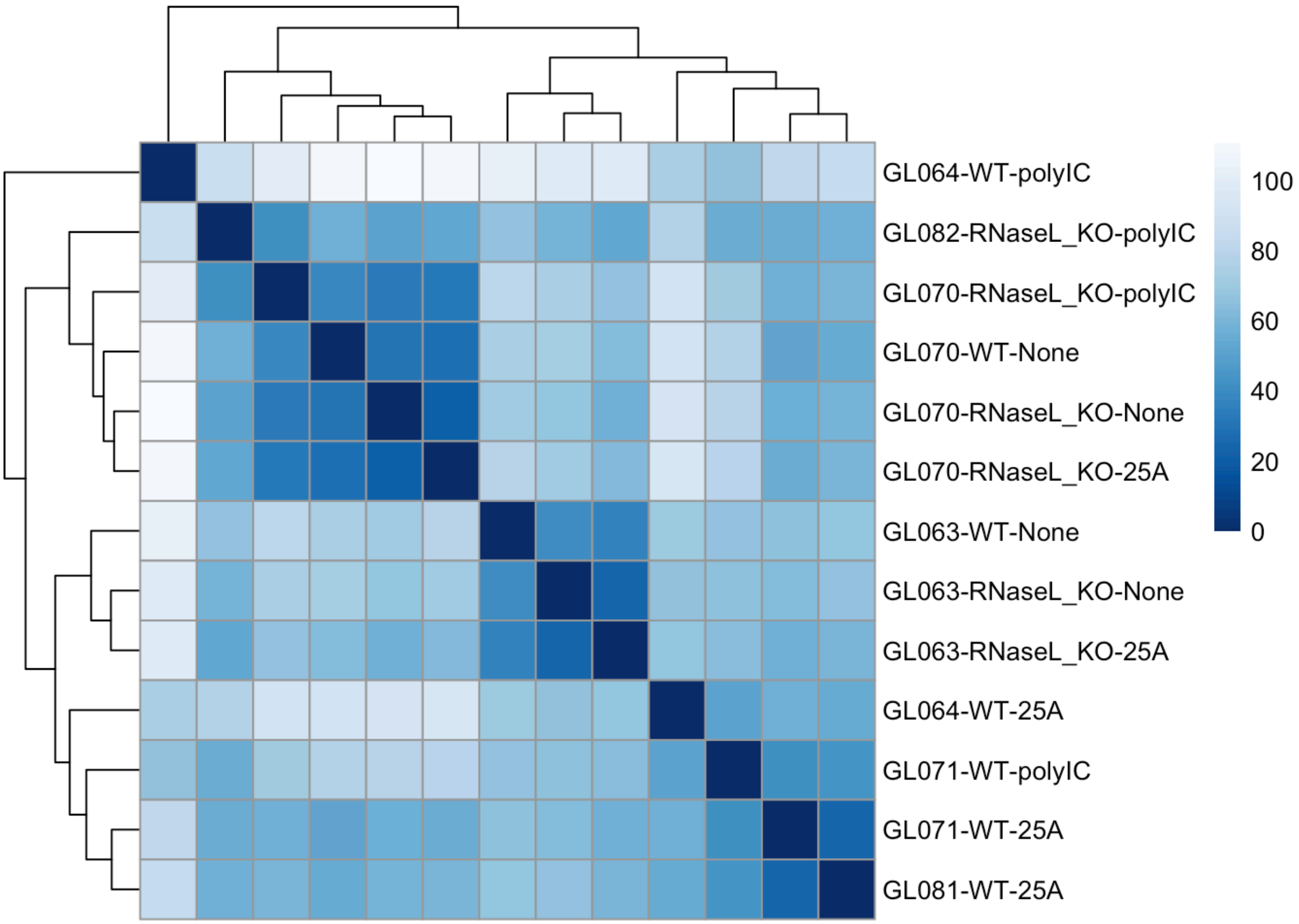
```
# add new factors (Group_gt_ind and Read_depth (high > 10M reads / Low < 10M read
meta_one_RS$Group_gt_ind <- factor(paste0(meta_one_RS$Genotype, meta_one_RS$Induce
meta_one_RS$Read_depth <- 'High'
meta_one_RS[meta_one_RS$Read_counts < 10e6,]$Read_depth <- 'Low'
meta_one_RS$Read_depth <- as.factor(meta_one_RS$Read_depth)

# Adding read_depth in design to control for read_depth
dds.one.RS <- DESeqDataSetFromMatrix(countData = all_one_RS,
                              colData = meta_one_RS,
                              design = ~ Read_depth + Group_gt_ind)
```

Exploratory analysis with DESeq object- RIBOseq

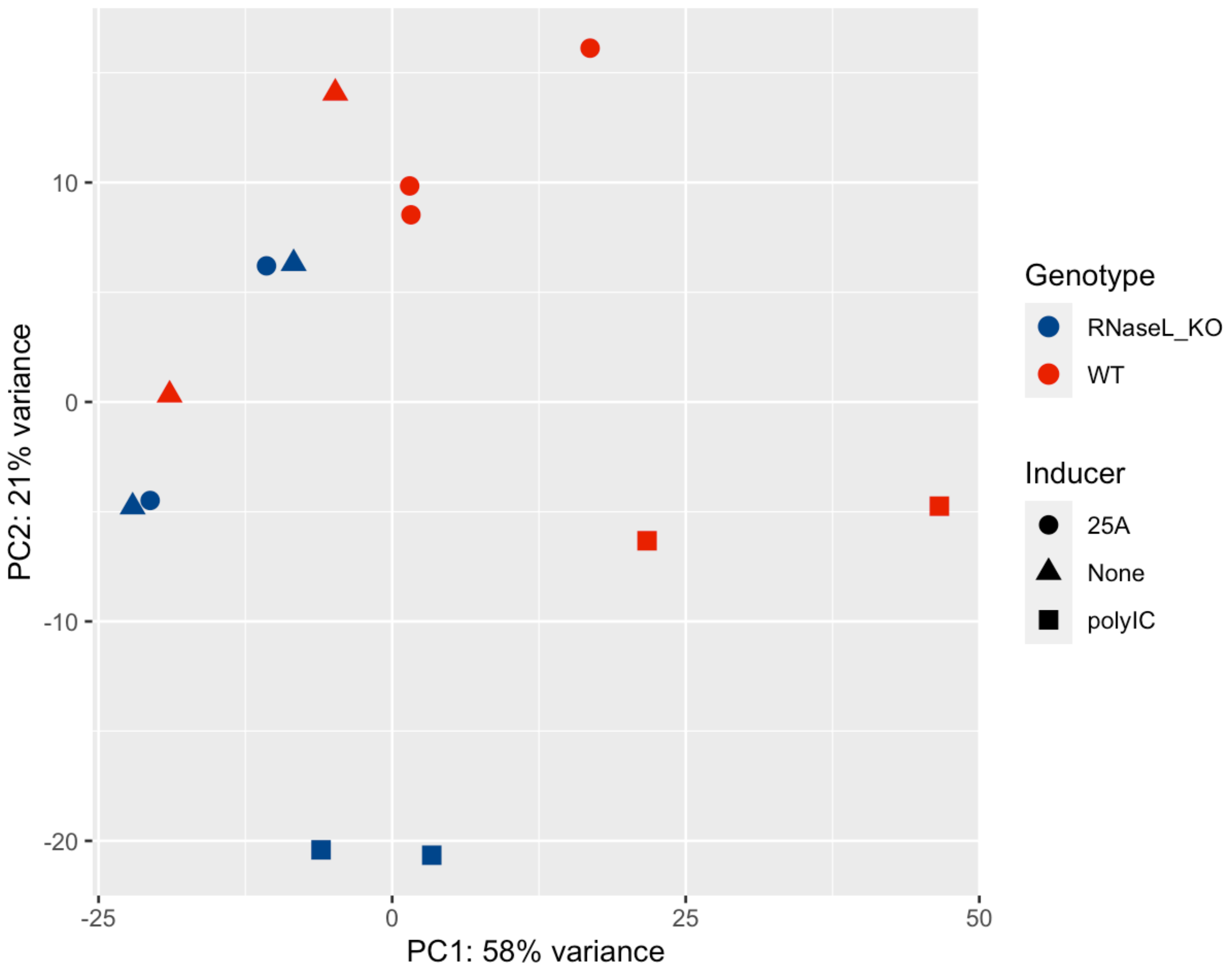
```
# Plot total reads per sample using barchar
p <- ggbarplot(data = meta_one_RS,
          x = "Sample_id",
          y = "Read_counts",
          x.text.angle = 90,
          fill = "Inducer",
          title = "Total read counts",
          ylab = "Read counts",
          sort.by.groups = TRUE,
          palette = "jco",
          sort.val = "asc",
          facet.by = "Genotype")
ggsave("Plots/barplot_read_counts_RS.pdf", plot = p)
p
```



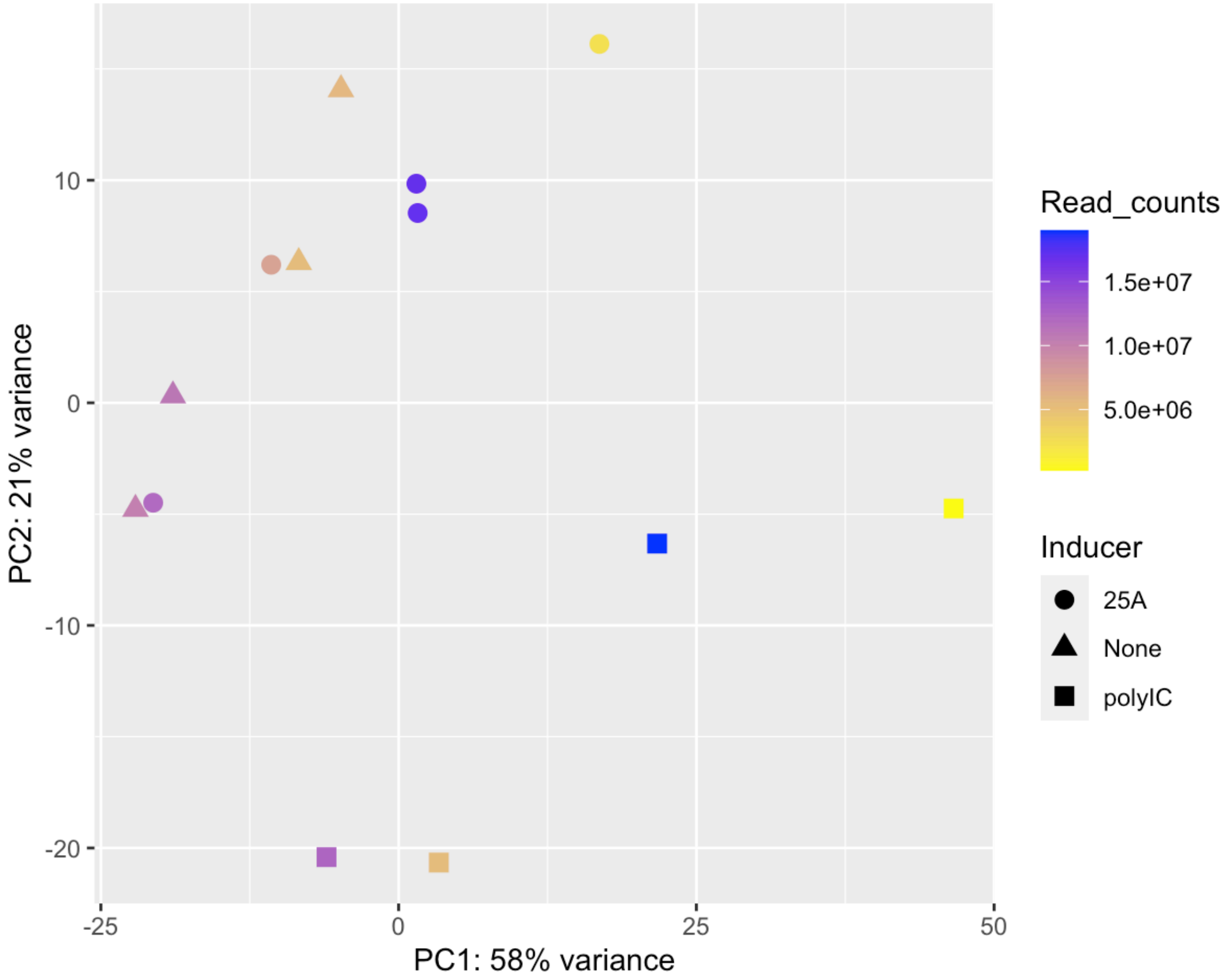
```
# Normalize counts
rlog.one <- rlog(dds.one.RS, blind=FALSE)

# Keep genes with at least 20 reads total across samples
keep <- rowSums(counts(dds.one.RS)) >= 0 # Agnes wanted to keep all genes for the
dds.one.RS <- dds.one.RS[keep,]

# Calculate distances between samples
sampleDists <- dist(t(assay(rlog.one)))

# Plot inter-sample distances
old.par <- par(no.readonly=T)

sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- paste(rlog.one$Sequencing_pool, rlog.one$Genotype,
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors)
```



```
# PCA
my_top_genes = 500

pcaData <- plotPCA(rlog.one, intgroup=c("Genotype", "Inducer"), returnData=TRUE, r
percentVar <- round(100 * attr(pcaData, "percentVar"))
y.coords = c(min(pcaData$PC1, pcaData$PC2), max(pcaData$PC1, pcaData$PC2))
x.coords = y.coords
p1 <- ggplot(pcaData, aes(PC1, PC2, color=Genotype, shape=Inducer)) +
  geom_point(size=3) + scale_color_lancet() +
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance")) +
  coord_fixed(ratio = (max(pcaData$PC1)-min(pcaData$PC1))/(max(pcaData$PC2)-min(pc

ggsave("Plots/pca_dataset_1_Induc_gt_RS.pdf", plot = p1)
p1
```



```
pcaData <- plotPCA(rlog.one, intgroup=c("Read_counts", "Inducer"), returnData=TRUE
percentVar <- round(100 * attr(pcaData, "percentVar"))
p2 <- ggplot(pcaData, aes(PC1, PC2, color=Read_counts, shape=Inducer)) +
  geom_point(size=3) +
  xlab(paste0("PC1: ",percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",percentVar[2],"% variance")) +
  coord_fixed(ratio = (max(pcaData$PC1)-min(pcaData$PC1))/(max(pcaData$PC2)-min(pc

ggsave("Plots/pca_dataset_1_Induc_read_counts_RS.pdf", plot = p2)
p2
```



PCA plots indicate that samples separated on PC1 mostly by sequencing depth, within treatments and genotypes. This implies that sequencing depth has to be controlled for by including this factor in the design formula.

Filtering out poorly-expressed genes (less than 20 reads across all samples) - RIBOseq

```
# Keep genes with at least 10 reads total across samples
keep <- rowSums(counts(dds.one.RS)) >= 0 # Agnes wanted to keep all genes
dds.one.RS <- dds.one.RS[keep,]
```

Spliting DESeq object based on genotype - RIBOseq

```
dds.one.RS.wt <- dds.one.RS[ , dds.one.RS$Genotype == "WT"]
dds.one.RS.wt$Genotype <- droplevels( dds.one.RS.wt$Genotype)
dds.one.RS.wt$Group_gt_ind <- droplevels( dds.one.RS.wt$Group_gt_ind)
dds.one.RS.wt$Group <- droplevels( dds.one.RS.wt$Group)

dds.one.RS.ko <- dds.one.RS[ , dds.one.RS$Genotype == "RNaseL_KO"]
dds.one.RS.ko$Genotype <- droplevels( dds.one.RS.ko$Genotype)
dds.one.RS.ko$Group_gt_ind <- droplevels( dds.one.RS.ko$Group_gt_ind)
dds.one.RS.ko$Group <- droplevels( dds.one.RS.ko$Group)
```

Calculate differential expression for WT - RIBOseq

```
# Calculate DE for WT samples
dds.one.RS.wt$Group_gt_ind <- relevel(dds.one.RS.wt$Group_gt_ind, "WTNone")
dds.one.RS.wt <- DESeq(dds.one.RS.wt)
resultsNames(dds.one.RS.wt)
```

```
## [1] "Intercept"                        "Read_depth_Low_vs_High"
## [3] "Group_gt_ind_WT25A_vs_WTNone"     "Group_gt_ind_WTpolyIC_vs_WTNone"
```

```
# Using lfcShrink instead of results to reduce high Log2FC bias of genes with low
res_wtIC_vs_wtNone <- lfcShrink(dds.one.RS.wt, coef = "Group_gt_ind_WTpolyIC_vs_W
res_wt25A_vs_wtNone <- lfcShrink(dds.one.RS.wt, coef = "Group_gt_ind_WT25A_vs_WTN

summary(res_wtIC_vs_wtNone, alpha = 0.05)
```

```
## 
## out of 17311 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 1164, 6.7%
## LFC < 0 (down)     : 935, 5.4%
## outliers [1]       : 0, 0%
## low counts [2]     : 3965, 23%
## (mean count < 3)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
summary(res_wt25A_vs_wtNone, alpha = 0.05)
```

```
##
## out of 17311 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 653, 3.8%
## LFC < 0 (down)     : 338, 2%
## outliers [1]       : 0, 0%
## low counts [2]     : 4625, 27%
## (mean count < 5)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Calculate differential expression for RNaseL_KO - RIBOseq

```
dds.one.RS.ko$Group_gt_ind <- relevel(dds.one.RS.ko$Group_gt_ind, "RNaseL_KONone")
design(dds.one.RS.ko) <- ~Group_gt_ind # Changing design given that there is no K
# Error: full model matrix is less than full rank
dds.one.RS.ko <- DESeq(dds.one.RS.ko)
resultsNames(dds.one.RS.ko)
```

```
## [1] "Intercept"
## [2] "Group_gt_ind_RNaseL_KO25A_vs_RNaseL_KONone"
## [3] "Group_gt_ind_RNaseL_KOpolyIC_vs_RNaseL_KONone"
```

```
res_koIC_vs_koNone <- lfcShrink(dds.one.RS.ko, coef = "Group_gt_ind_RNaseL_KOpolyI
res_ko25A_vs_koNone <- lfcShrink(dds.one.RS.ko, coef = "Group_gt_ind_RNaseL_KO25A_

summary(res_koIC_vs_koNone, alpha = 0.05)
```

```
##
## out of 16151 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 51, 0.32%
## LFC < 0 (down)     : 0, 0%
## outliers [1]       : 0, 0%
## low counts [2]     : 2760, 17%
## (mean count < 2)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
summary(res_ko25A_vs_koNone, alpha = 0.05)
```

```
##
## out of 16151 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 0, 0%
## LFC < 0 (down)     : 0, 0%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Write DE tables to file - exons

```
# Sort results by Log2FC
res_wtIC_vs_wtNone.logfc_sorted <- sort_and_write_res_table(res_wtIC_vs_wtNone, "
res_koIC_vs_koNone.logfc_sorted <- sort_and_write_res_table(res_koIC_vs_koNone, "
res_wt25A_vs_wtNone.logfc_sorted <- sort_and_write_res_table(res_wt25A_vs_wtNone,
res_ko25A_vs_koNone.logfc_sorted <- sort_and_write_res_table(res_ko25A_vs_koNone,

# Save sorted files as a list
DE_results[["wtIC_vs_wtNone_RS"]]  <- res_wtIC_vs_wtNone.logfc_sorted
DE_results[["koIC_vs_koNone_RS"]] <- res_koIC_vs_koNone.logfc_sorted
DE_results[["wt25A_vs_wtNone_RS"]] <- res_wt25A_vs_wtNone.logfc_sorted
DE_results[["ko25A_vs_koNone_RS"]] <- res_ko25A_vs_koNone.logfc_sorted
```

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Ventura 13.2.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRb
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRla
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] cowplot_1.1.1               RColorBrewer_1.1-3
##  [3] viridis_0.6.3               viridisLite_0.4.2
##  [5] ggsci_3.0.0                 pcaExplorer_2.24.0
##  [7] biomaRt_2.54.1              stringr_1.5.0
##  [9] DESeq2_1.38.3               SummarizedExperiment_1.28.0
## [11] MatrixGenerics_1.10.0       matrixStats_1.0.0
## [13] GenomicRanges_1.50.2        GenomeInfoDb_1.34.9
## [15] ggpubr_0.6.0                ggplot2_3.4.2
## [17] pheatmap_1.0.12             org.Hs.eg.db_3.16.0
## [19] AnnotationDbi_1.60.2        IRanges_2.32.0
## [21] S4Vectors_0.36.2            Biobase_2.58.0
## [23] BiocGenerics_0.44.0
##
## loaded via a namespace (and not attached):
##   [1] GOstats_2.64.0         backports_1.4.1        systemfonts_1.0.4
##   [4] BiocFileCache_2.6.1    NMF_0.26               plyr_1.8.8
##   [7] igraph_1.5.0           lazyeval_0.2.2         GSEABase_1.60.0
##  [10] shinydashboard_0.7.2   splines_4.2.2          BiocParallel_1.32.6
##  [13] crosstalk_1.2.0        gridBase_0.4-7         digest_0.6.33
##  [16] invgamma_1.1           ca_0.71.1              foreach_1.5.2
##  [19] htmltools_0.5.5        GO.db_3.16.0           SQUAREM_2021.1
##  [22] fansi_1.0.4            magrittr_2.0.3         memoise_2.0.1
##  [25] cluster_2.1.4          doParallel_1.0.17      limma_3.54.2
##  [28] Biostrings_2.66.0      annotate_1.76.0        prettyunits_1.1.1
##  [31] colorspace_2.1-0       blob_1.2.4             rappdirs_0.3.3
##  [34] ggrepel_0.9.3          textshaping_0.3.6      xfun_0.39
##  [37] dplyr_1.1.2            crayon_1.5.2           RCurl_1.98-1.12
##  [40] jsonlite_1.8.7         graph_1.76.0           genefilter_1.80.3
##  [43] survival_3.4-0         iterators_1.0.14       glue_1.6.2
##  [46] registry_0.5-1         gtable_0.3.3           zlibbioc_1.44.0
##  [49] XVector_0.38.0         webshot_0.5.5          DelayedArray_0.24.0
##  [52] car_3.1-2              Rgraphviz_2.42.0       abind_1.4-5
##  [55] SparseM_1.81           scales_1.2.1           DBI_1.1.3
##  [58] rngtools_1.5.2         rstatix_0.7.2          Rcpp_1.0.11
##  [61] xtable_1.8-4           progress_1.2.2         bit_4.0.5
##  [64] DT_0.28                truncnorm_1.0-9        AnnotationForge_1.40.2
##  [67] htmlwidgets_1.6.2      httr_1.4.6             threejs_0.3.3
##  [70] shinyAce_0.4.2         ellipsis_0.3.2         farver_2.1.1          
##  [73] pkgconfig_2.0.3        XML_3.99-0.14          sass_0.4.6            
##  [76] dbplyr_2.3.3           locfit_1.5-9.8         utf8_1.2.3            
##  [79] labeling_0.4.2         tidyselect_1.2.0       rlang_1.1.1           
##  [82] reshape2_1.4.4         later_1.3.1            munsell_0.5.0         
##  [85] tools_4.2.2            cachem_1.0.8           cli_3.6.1             
##  [88] generics_0.1.3         RSQLite_2.3.1          broom_1.0.5           
##  [91] shinyBS_0.61.1         evaluate_0.21          fastmap_1.1.1         
##  [94] ragg_1.2.5             heatmaply_1.4.2        yaml_2.3.7            
##  [97] knitr_1.43             bit64_4.0.5            purrr_1.0.1           
## [100] KEGGREST_1.38.0        dendextend_1.17.1      RBGL_1.74.0           
## [103] mime_0.12              xml2_1.3.5             compiler_4.2.2        
## [106] rstudioapi_0.15.0      plotly_4.10.2          filelock_1.0.2        
## [109] curl_5.0.1             png_0.1-8              ggsignif_0.6.4        
## [112] tibble_3.2.1           geneplotter_1.76.0     bslib_0.5.0           
## [115] stringi_1.7.12         highr_0.10             lattice_0.20-45       
## [118] Matrix_1.5-4.1         vctrs_0.6.3            pillar_1.9.0          
## [121] lifecycle_1.0.3        BiocManager_1.30.21    jquerylib_0.1.4       
## [124] irlba_2.3.5.1          data.table_1.14.8      bitops_1.0-7          
## [127] seriation_1.4.2        httpuv_1.6.11          R6_2.5.1              
## [130] TSP_1.2-4              promises_1.2.0.1       topGO_2.50.0          
## [133] gridExtra_2.3          codetools_0.2-18       assertthat_0.2.1      
## [136] Category_2.64.0        withr_2.5.0            GenomeInfoDbData_1.2.9
## [139] parallel_4.2.2         hms_1.1.3              grid_4.2.2            
## [142] tidyr_1.3.0            rmarkdown_2.23         ashr_2.2-54           
## [145] carData_3.0-5          mixsqp_0.3-48          shiny_1.7.4.1         
## [148] base64enc_0.1-3
```