CYTO DE

Barry Digby

30/11/2018

Report Outline:

This report is on Differential Expression (DE) between S2 and WT samples in the Control samples. # Stringtie to DESeq2 object:

```
library(DESeq2)
```

```
## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colMeans, colnames, colSums, dirname, do.call, duplicated,
##     eval, evalq, Filter, Find, get, grep, grepl, intersect,
##     is.unsorted, lapply, lengths, Map, mapply, match, mget, order,
##     paste, pmax, pmax.int, pmin, pmin.int, Position, rank, rbind,
##     Reduce, rowMeans, rownames, rowSums, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
##     which.min

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Loading required package: SummarizedExperiment
```

```
## Loading required package: Biobase
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
## Loading required package: DelayedArray
```

```
## Loading required package: matrixStats
```

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

```
## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians
```

```
## Loading required package: BiocParallel
```

```
##
## Attaching package: 'DelayedArray'
```

```
## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges
```

```
## The following objects are masked from 'package:base':
##
##     aperm, apply
```

```
library(tximport)

directory <- "/Users/barrydigby/Desktop/read_tables/"

data <- c("/Users/barrydigby/Desktop/read_tables/H84S2Ctrl_S7/t_data.ctab",
          "/Users/barrydigby/Desktop/read_tables/H84WTCtrl_S1/t_data.ctab",
          "/Users/barrydigby/Desktop/read_tables/H85S2Ctrl_S8/t_data.ctab",
          "/Users/barrydigby/Desktop/read_tables/H85WTCtrl_S2/t_data.ctab",
          "/Users/barrydigby/Desktop/read_tables/H86S2Ctrl_S9/t_data.ctab",
          "/Users/barrydigby/Desktop/read_tables/H86WTCtrl_S3/t_data.ctab")

tmp <- read.table(data[1], header = TRUE)
names(data) <- c("S2Ctrl_7", "WTCtrl_1", "S2Ctrl_8", "WTCtrl_2", "S2Ctrl_9", "WTCtrl_3")
tx2gene <- tmp[, c("t_name", "gene_name")]
txi <- tximport(data, type = "stringtie", tx2gene = tx2gene)
```

```
## reading in files with read_tsv
```

```
## 1 2 3 4 5 6
## summarizing abundance
## summarizing counts
## summarizing length
```

```
sampleNames <- c("S2Ctrl_7", "WTCtrl_1", "S2Ctrl_8", "WTCtrl_2", "S2Ctrl_9", "WTCtrl_3")
sampleGroup <- c("S2", "WT", "S2", "WT", "S2", "WT")
sampleTable <- data.frame(sampleName = sampleNames, type = sampleGroup)
```

```
rownames(sampleTable) <- colnames(txi$counts)
ddsTxi <- DESeqDataSetFromTximport(txi, sampleTable, design = ~ type)
```

```
## using counts and average transcript lengths from tximport
```

```
dds <- DESeq(ddsTxi)
```

```
## estimating size factors
## using 'avgTxLength' from assays(dds), correcting for library size
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
```

```
res <- results(dds)
counts <- counts(dds)
```

Data Quality Assessment

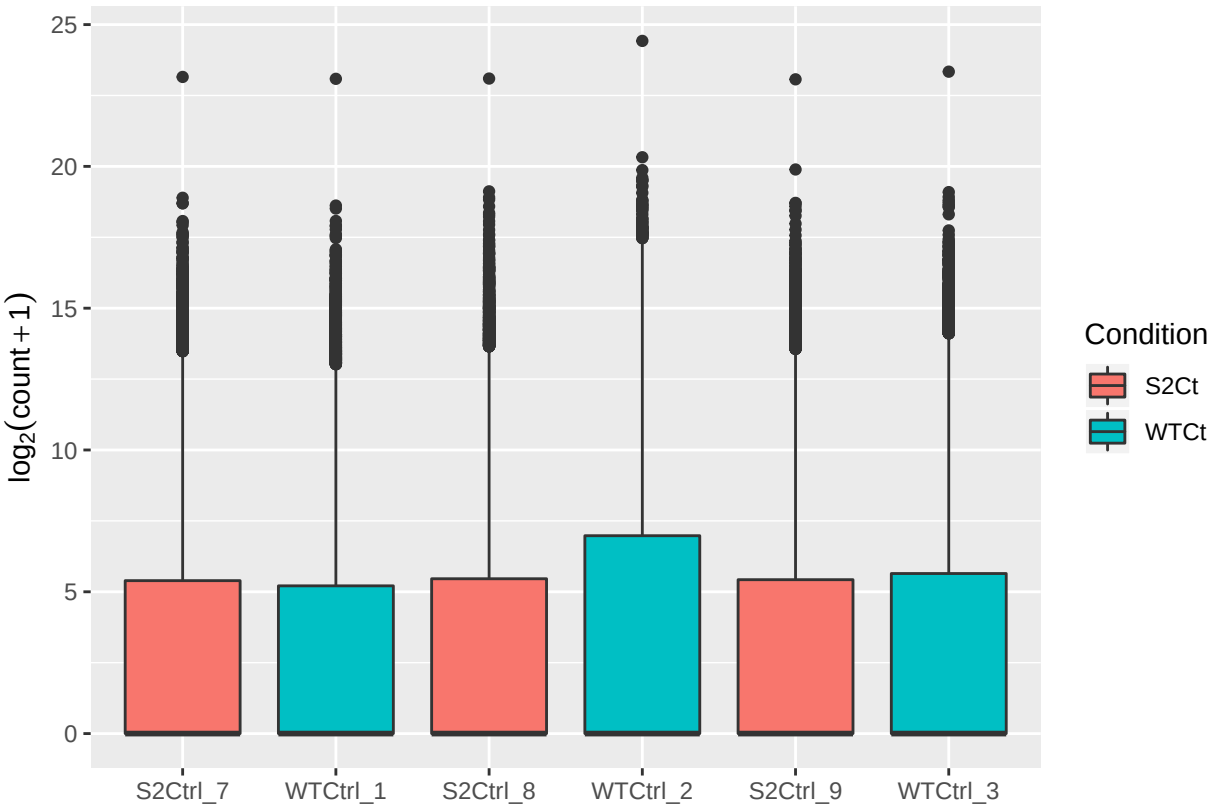
Transformation & Visualization

The majority of genes have zero or very low counts, thus we apply a log2-transformation on counts before visualization. The distribution of log2 transformed counts are shown as a boxplot and a density plot:

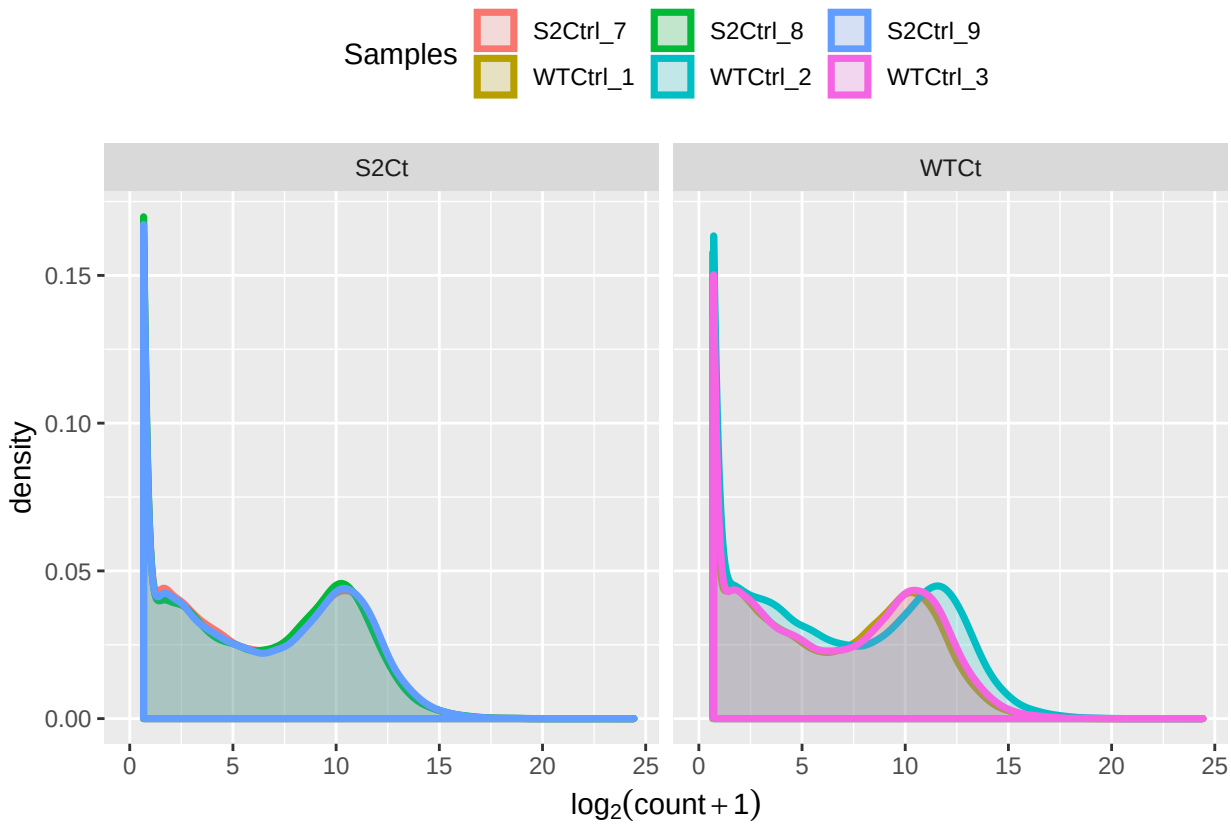
```
pseudoCount = log2(counts + 1)

library(reshape2)
library(ggplot2)
pseudoCount = as.data.frame(pseudoCount)
df = melt(pseudoCount, variable.name = "Samples", value.name = "count") # reshape the matrix
```

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

```
df = data.frame(df, Condition = substr(df$Samples, 1, 4))
ggplot(df, aes(x = df$Samples, y = count, fill = Condition)) + geom_boxplot() + xlab("") +
  ylab(expression(log[2](count + 1)))
```

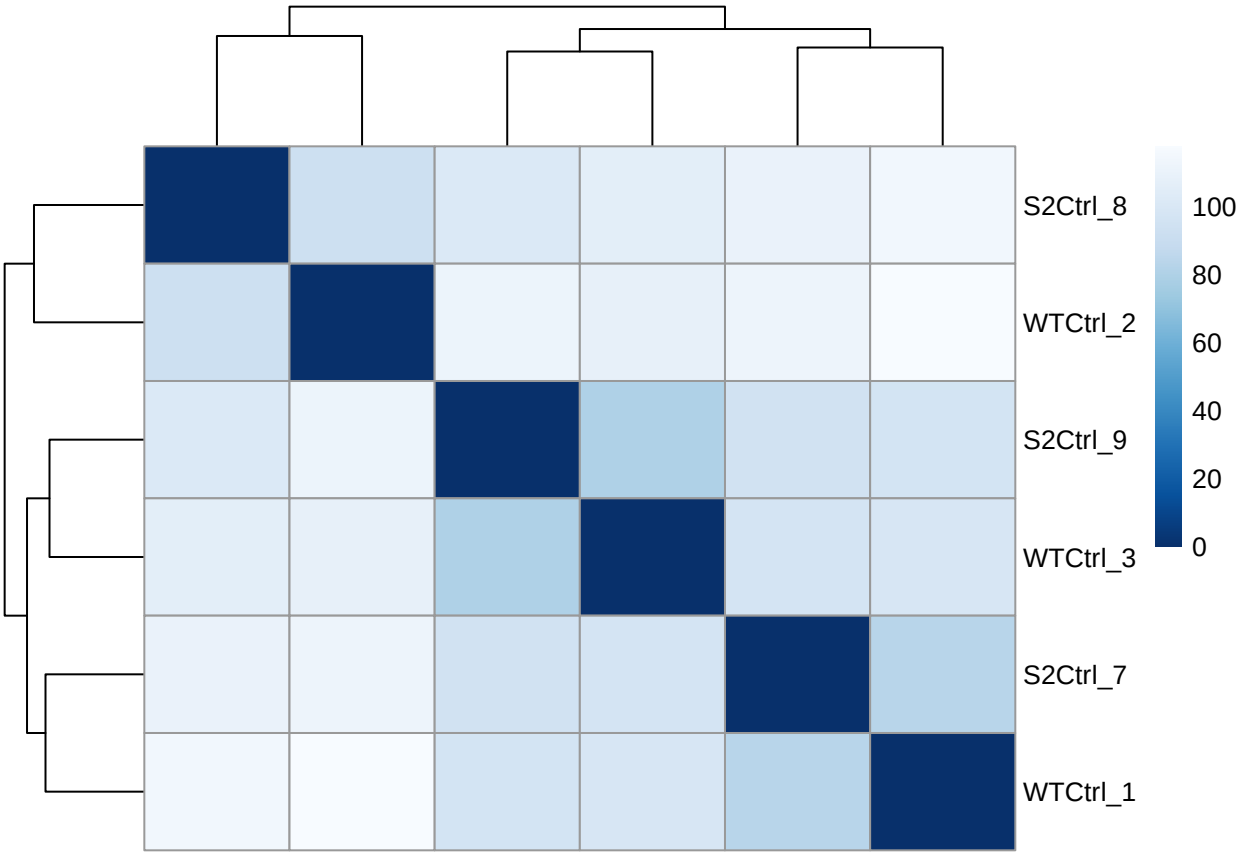



```
ggplot(df, aes(x = count, colour = Samples, fill = Samples)) + ylim(c(0, 0.17)) +
  geom_density(alpha = 0.2, size = 1.25) + facet_wrap(~ Condition) +
  theme(legend.position = "top") + xlab(expression(log[2](count + 1)))
```



Sample to Sample Distances

```
sampleDists <- dist(t(assay(vsd)))
library("pheatmap")
library("RColorBrewer")
sampleDistMatrix <- as.matrix(sampleDists)
rownames(sampleDistMatrix) <- vsd$sampleName
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors)
```



PCA plots

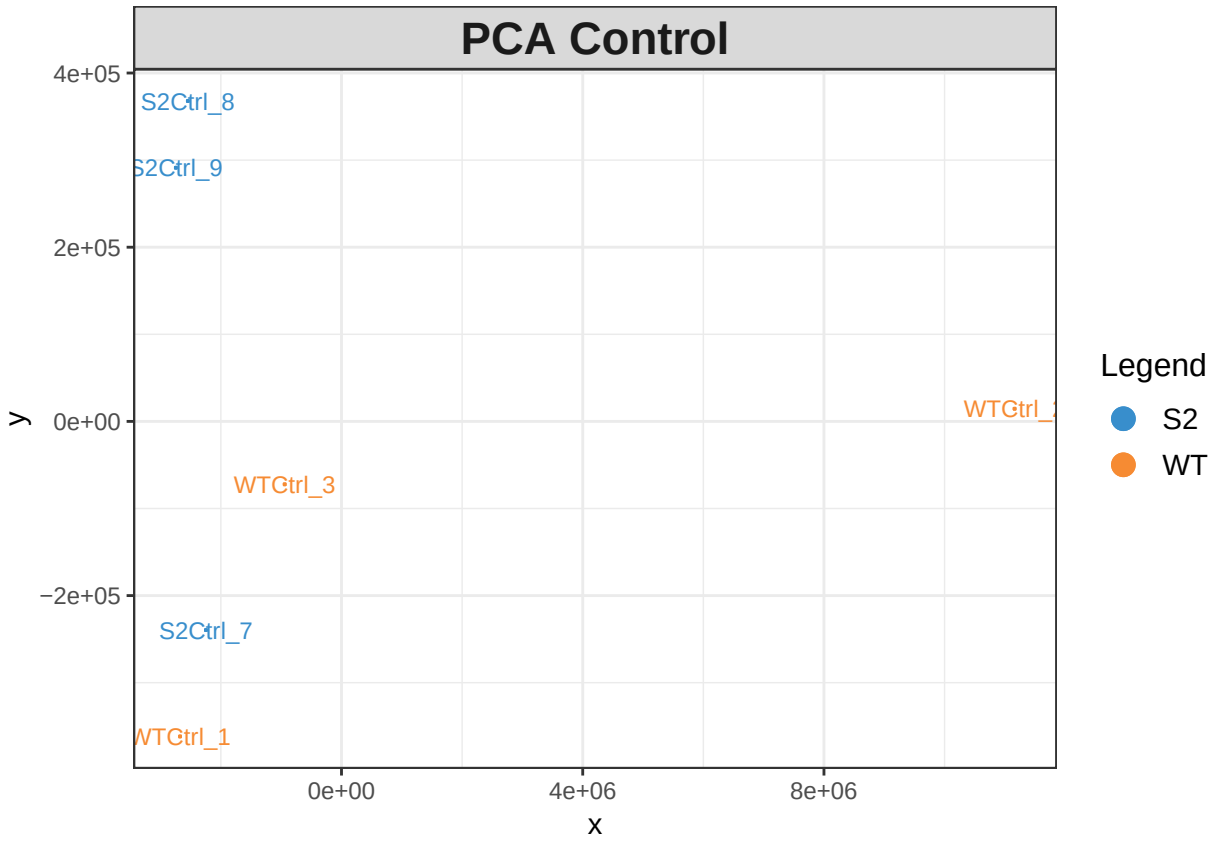
```
library(mixOmics)
```

```
## Loading required package: MASS
```

```
## Loading required package: lattice
```

```
##
## Loaded mixOmics 6.6.0
##
## Thank you for using mixOmics! Learn how to apply our methods with our tutorials on www.mixOmics.org,
## Questions: email us at mixomics[at]math.univ-toulouse.fr
## Bugs, Issues? https://github.com/mixOmicsTeam/mixOmics/issues
## Cite us:  citation('mixOmics')
```

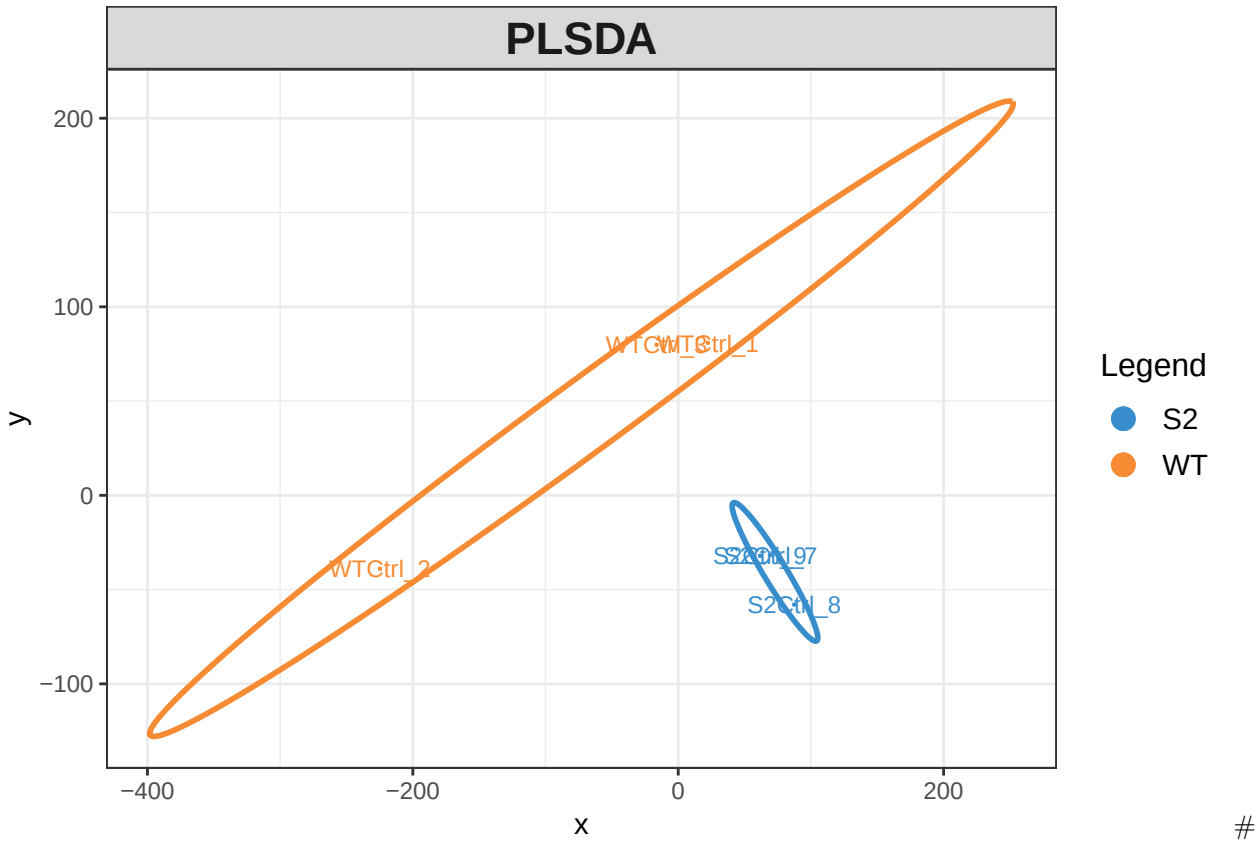
```
counts.t <- t(counts(dds))
outcome <- as.factor(dds$type)

pca = pca(counts.t, ncomp = 6)
plotIndiv(pca, group = outcome, ind.names = TRUE, legend = TRUE, title = 'PCA Control')
```




```
plsda_ <- plsda(counts.t, outcome, ncomp = 5)
```

```
## Warning in cor(A[[k]], variates.A[[k]]): the standard deviation is zero
```

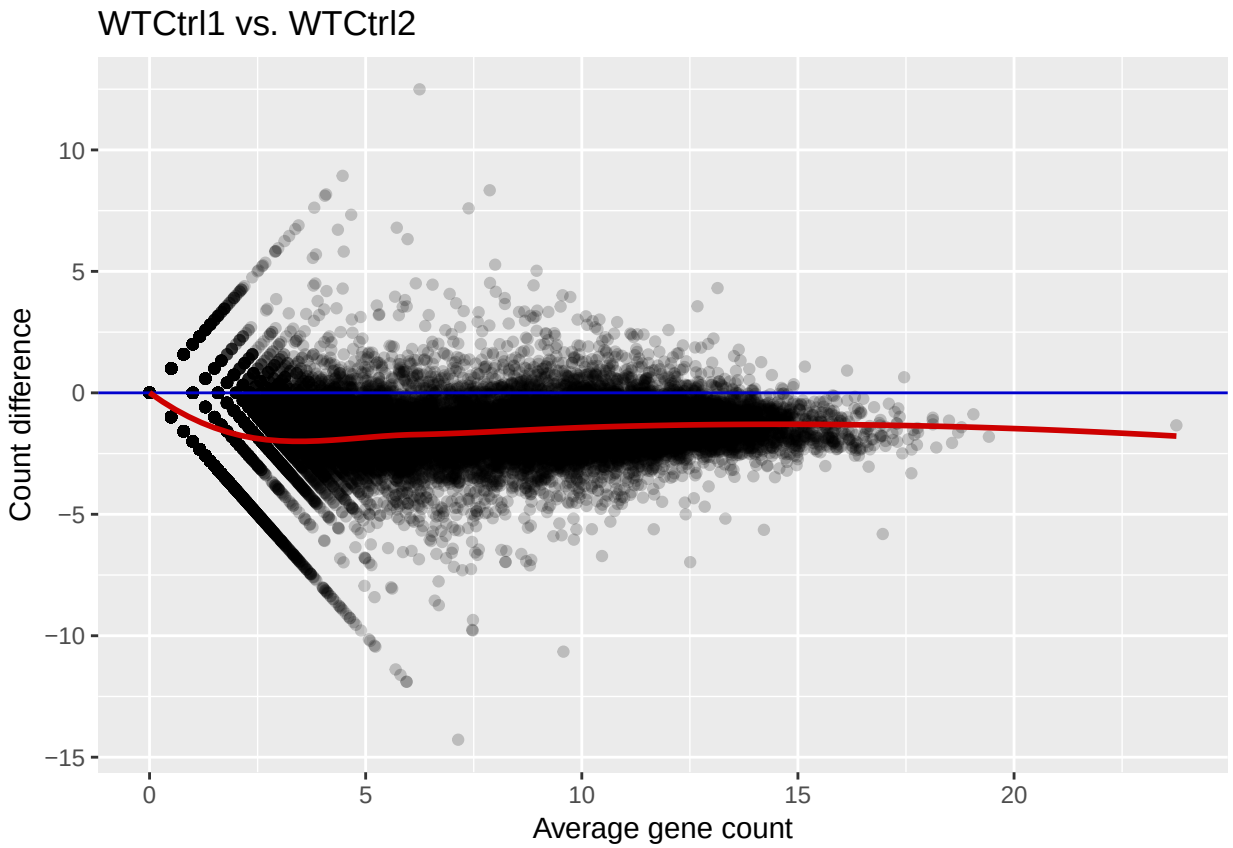
```
plotIndiv(plsda_, comp = 1:2, group = outcome, ind.names = TRUE, ellipse = TRUE, legend = TRUE, title =
```



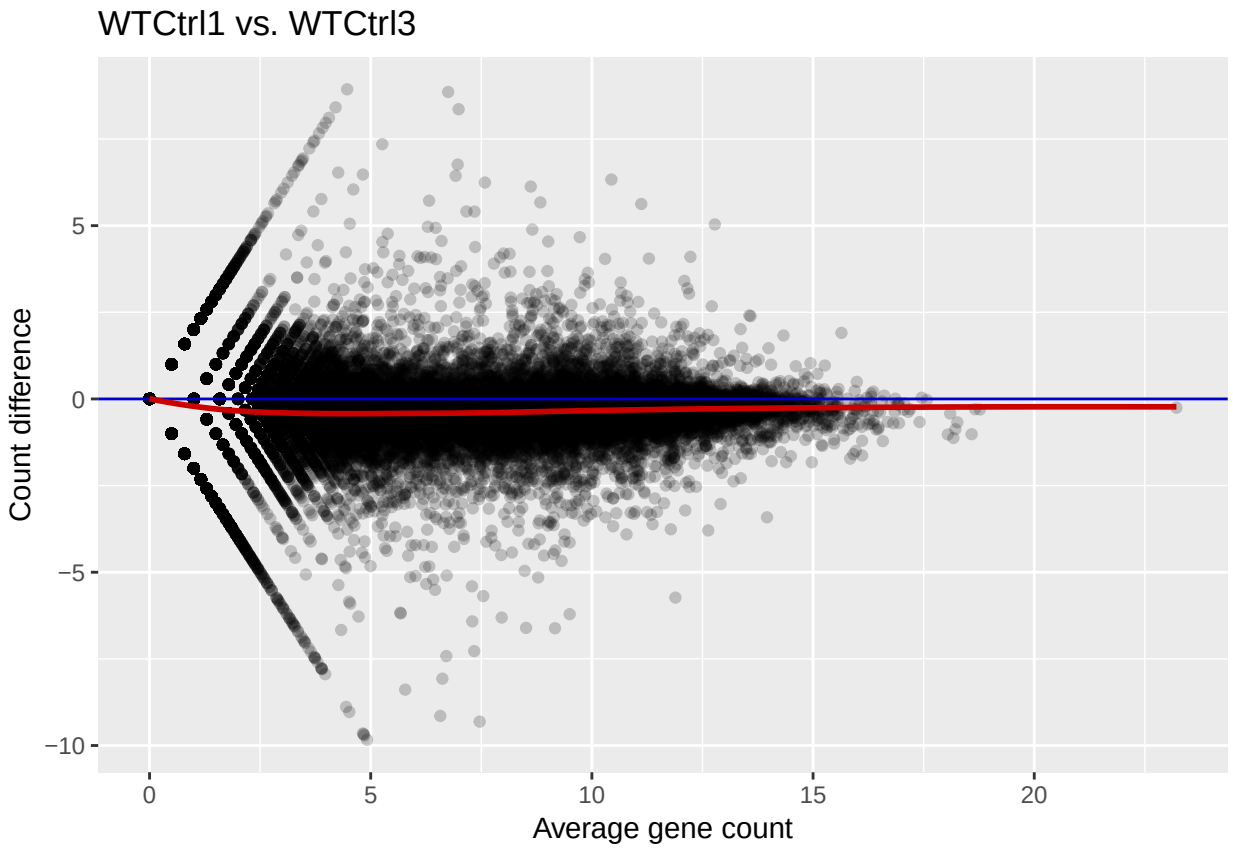
#

Systematic biases

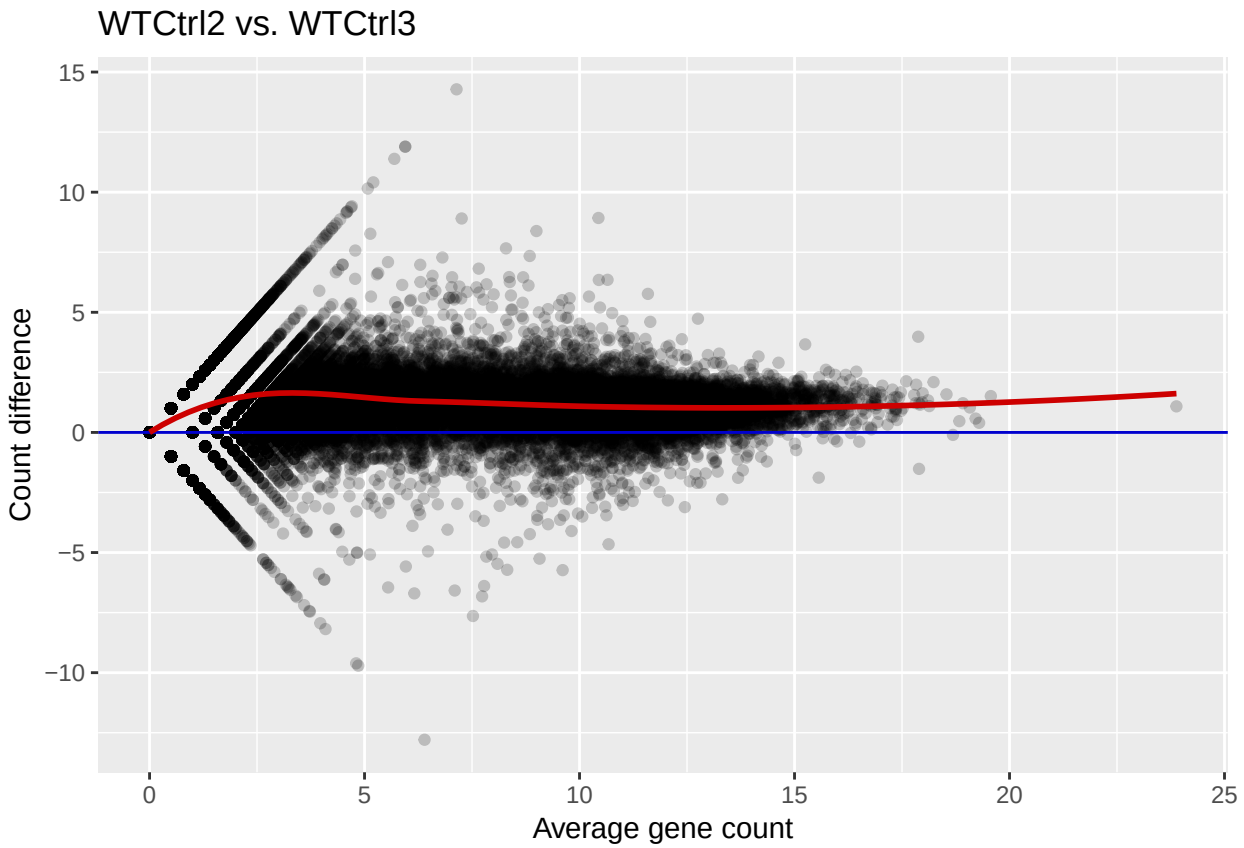
```
x = pseudoCount[, 2] #
y = pseudoCount[, 4]
M = x - y # M-values (differences between samples)
A = (x + y)/2 # A-values (averages)
df = data.frame(A, M)
ggplot(df, aes(x = A, y = M)) + geom_point(size = 1.5, alpha = 1/5) +
geom_hline(aes(yintercept = 0), color = "blue3") + stat_smooth(se =
FALSE, method = "loess", color = "red3") +
  xlab("Average gene count") +
  ylab("Count difference") +
  ggtitle("WTCtrl1 vs. WTCtrl2")
```



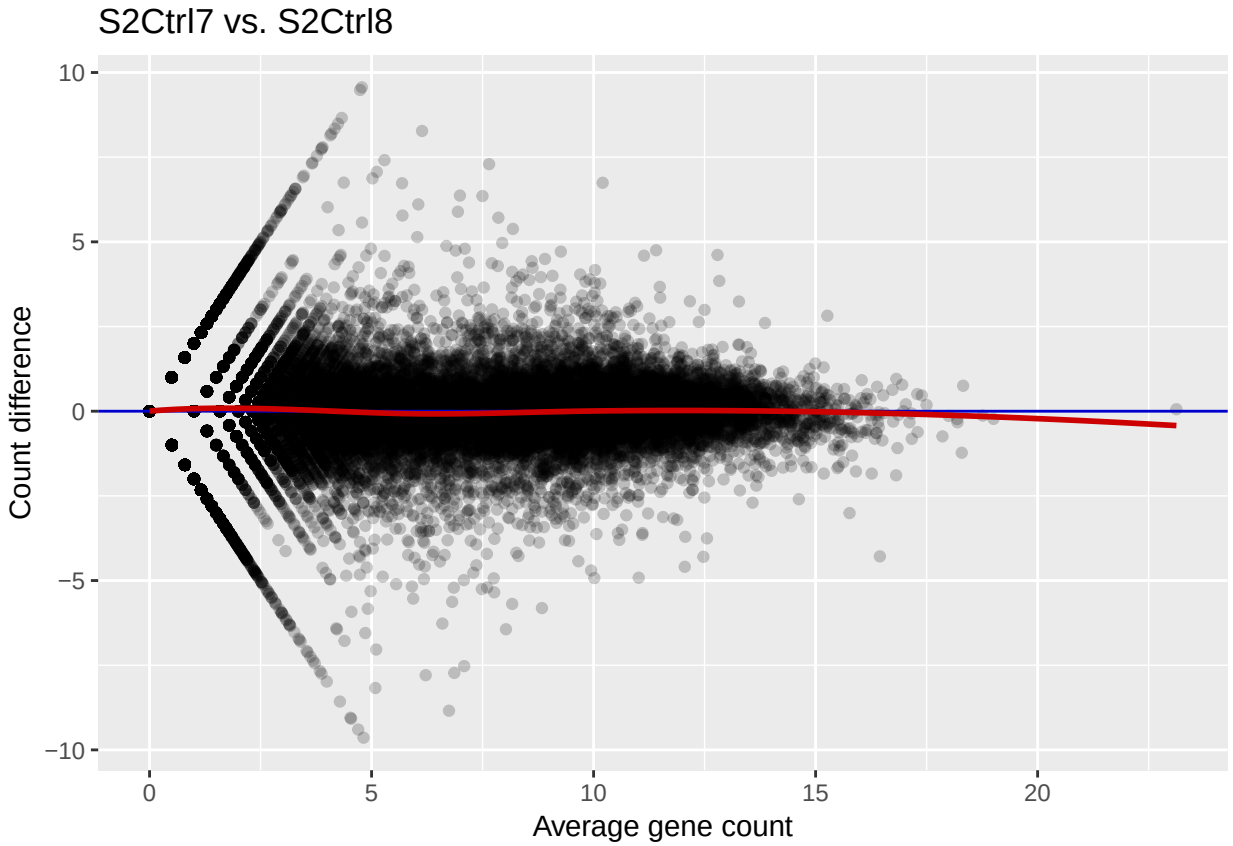

```
x = pseudoCount[, 2] #
y = pseudoCount[, 6] #
M = x - y # M-values (differences between samples)
A = (x + y)/2 # A-values (averages)
df = data.frame(A, M)
ggplot(df, aes(x = A, y = M)) + geom_point(size = 1.5, alpha = 1/5) +
geom_hline(aes(yintercept = 0), color = "blue3") + stat_smooth(se =
FALSE, method = "loess", color = "red3") +
  xlab("Average gene count") +
  ylab("Count difference") +
  ggtitle("WTCtrl1 vs. WTCtrl3")
```



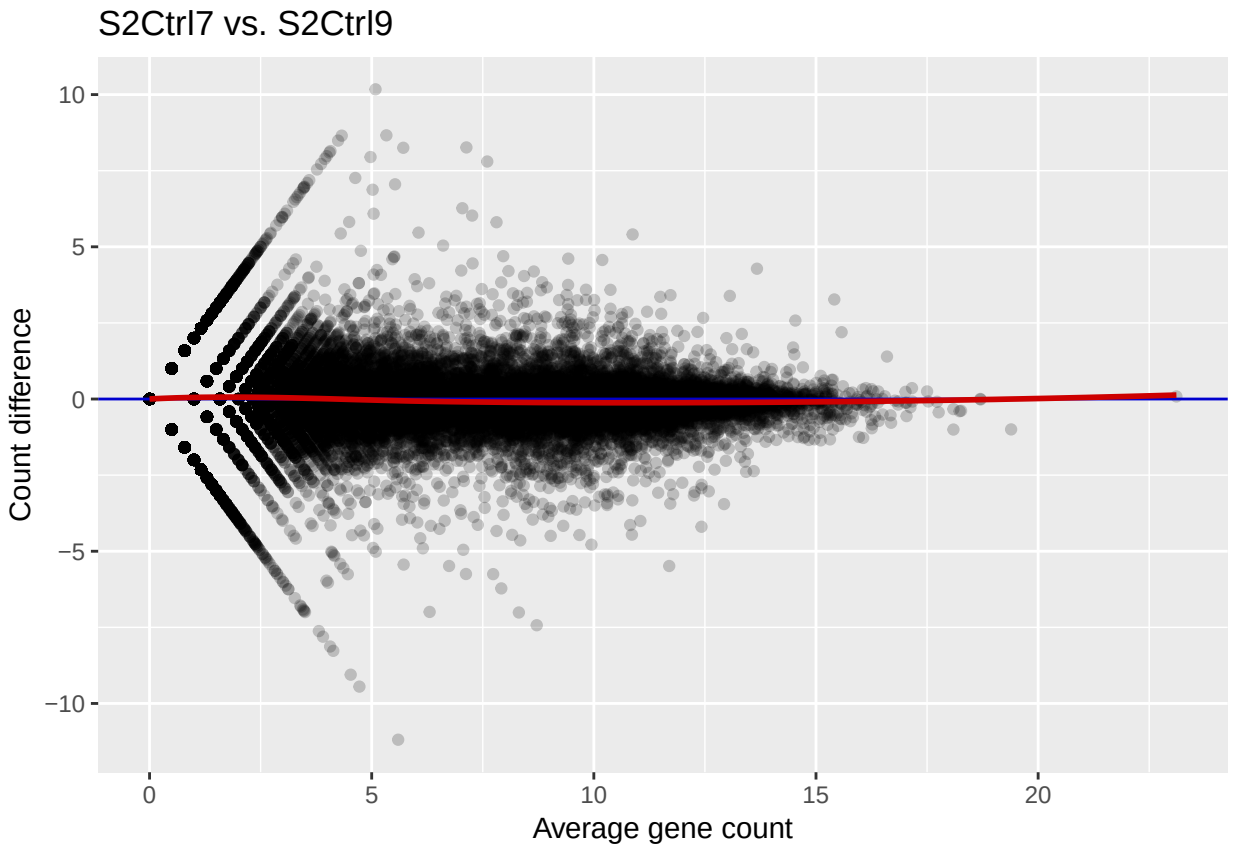

```
x = pseudoCount[, 4] # e
y = pseudoCount[, 6] #
M = x - y # M-values (differences between samples)
A = (x + y)/2 # A-values (averages)
df = data.frame(A, M)
ggplot(df, aes(x = A, y = M)) + geom_point(size = 1.5, alpha = 1/5) +
geom_hline(aes(yintercept = 0), color = "blue3") + stat_smooth(se =
FALSE, method = "loess", color = "red3") +
  xlab("Average gene count") +
  ylab("Count difference") +
  ggtitle("WTCtrl2 vs. WTCtrl3")
```



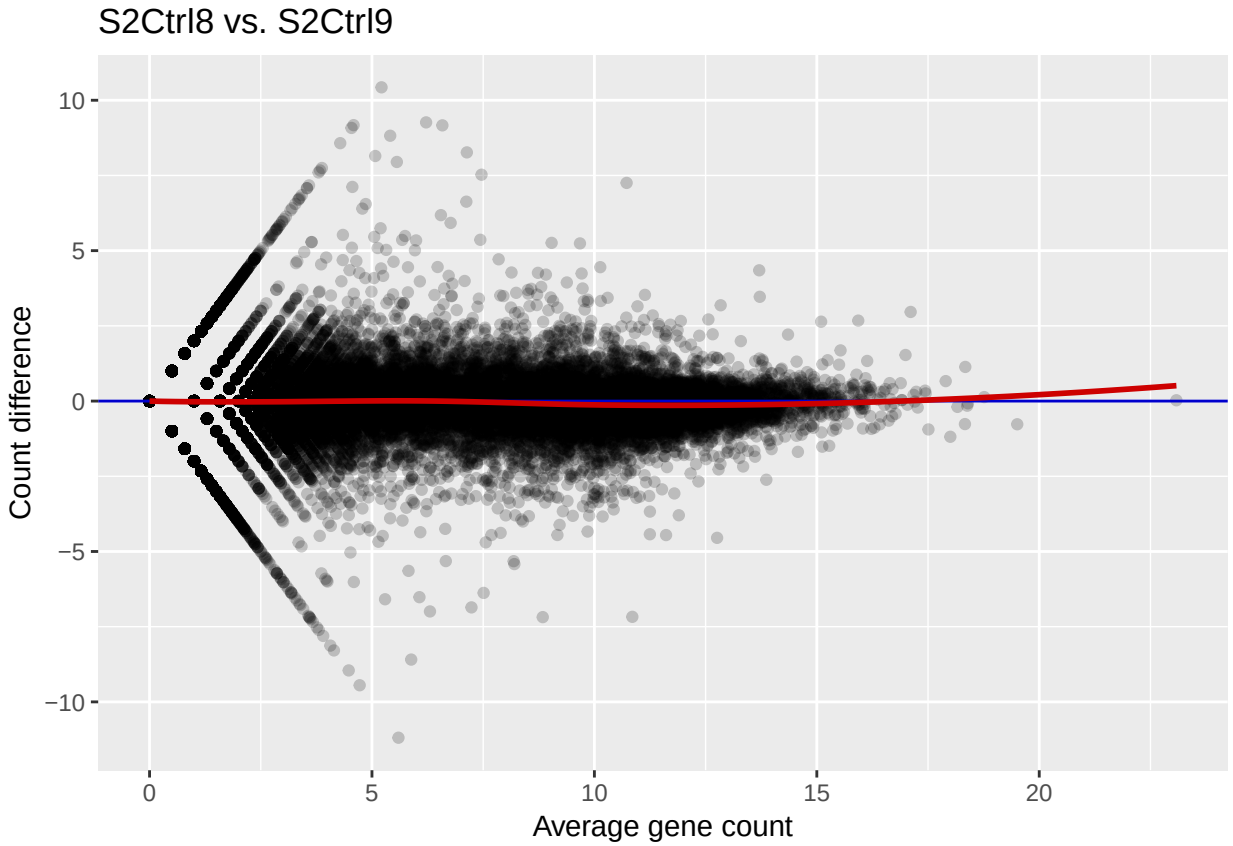

```
x = pseudoCount[, 1]
y = pseudoCount[, 3]
M = x - y # M-values (differences between samples)
A = (x + y)/2 # A-values (averages)
df = data.frame(A, M)
ggplot(df, aes(x = A, y = M)) + geom_point(size = 1.5, alpha = 1/5) +
geom_hline(aes(yintercept = 0), color = "blue3") + stat_smooth(se =
FALSE, method = "loess", color = "red3") +
  xlab("Average gene count") +
  ylab("Count difference") +
  ggtitle("S2Ctrl7 vs. S2Ctrl8")
```




```
x = pseudoCount[, 1]
y = pseudoCount[, 5]
M = x - y # M-values (differences between samples)
A = (x + y)/2 # A-values (averages)
df = data.frame(A, M)
ggplot(df, aes(x = A, y = M)) + geom_point(size = 1.5, alpha = 1/5) +
geom_hline(aes(yintercept = 0), color = "blue3") + stat_smooth(se =
FALSE, method = "loess", color = "red3") +
  xlab("Average gene count") +
  ylab("Count difference") +
  ggtitle("S2Ctrl7 vs. S2Ctrl9")
```




```
x = pseudoCount[, 3]
y = pseudoCount[, 5]
M = x - y # M-values (differences between samples)
A = (x + y)/2 # A-values (averages)
df = data.frame(A, M)
ggplot(df, aes(x = A, y = M)) + geom_point(size = 1.5, alpha = 1/5) +
geom_hline(aes(yintercept = 0), color = "blue3") + stat_smooth(se =
FALSE, method = "loess", color = "red3") +
  xlab("Average gene count") +
  ylab("Count difference") +
  ggtitle("S2Ctrl8 vs. S2Ctrl9")
```



Differential expression analysis

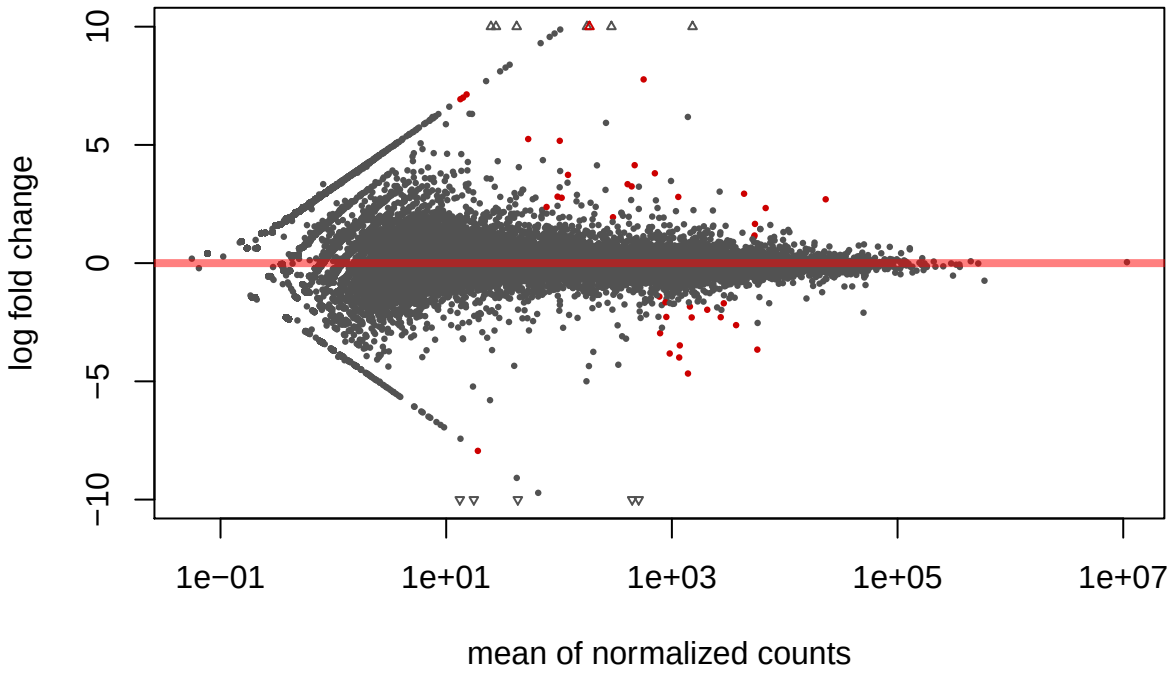
```
summary(res)
```

```
##
## out of 30845 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 22, 0.071%
## LFC < 0 (down)     : 16, 0.052%
## outliers [1]       : 117, 0.38%
## low counts [2]     : 9138, 30%
## (mean count < 3)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

MA Plot

Red points have adjusted p value < 0.1. We can see a lot of noise, using a shrunken log2 fold change reduces the noise.

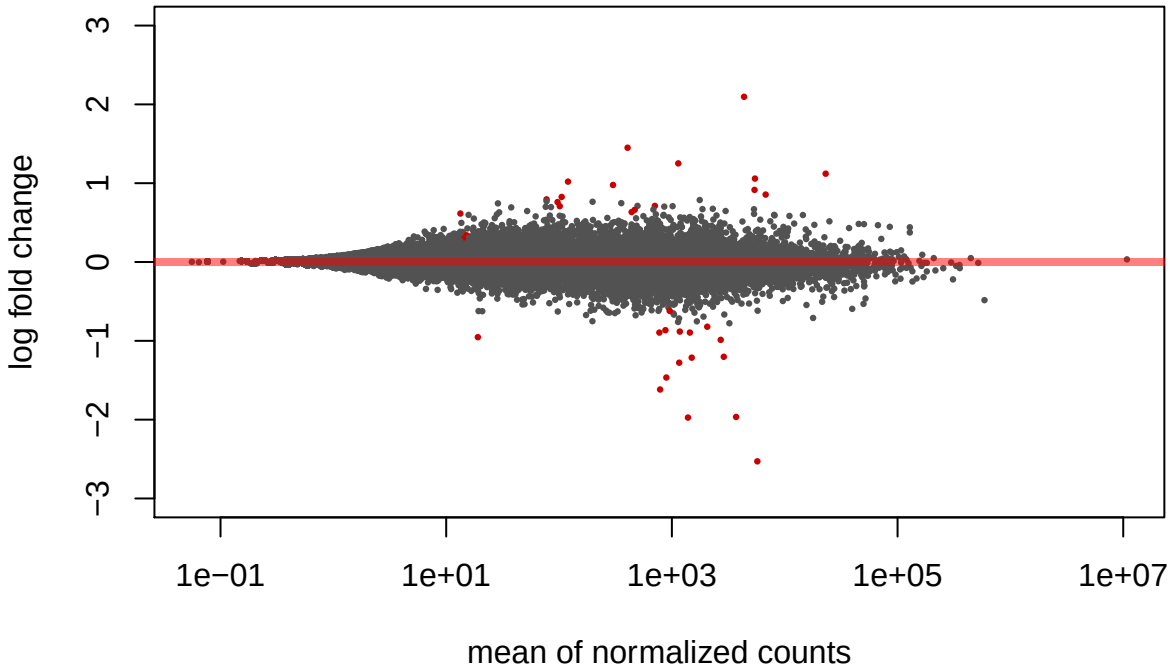
```
plotMA(res, ylim=c(-10,10))
```




```
resShrink = lfcShrink(dds, coef=2)
```

```
## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
## additional priors are available via the 'type' argument, see ?lfcShrink for details
```

```
plotMA(resShrink, ylim=c(-3,3))
```



Extract Significant genes: P value = 0.05 = 544 genes Adj P value = 0.1 = 38 genes Both write to csv ..

```
resOrdered <- res[order(res$pvalue),]

pvalue <- subset(resOrdered, pvalue < 0.05)
```

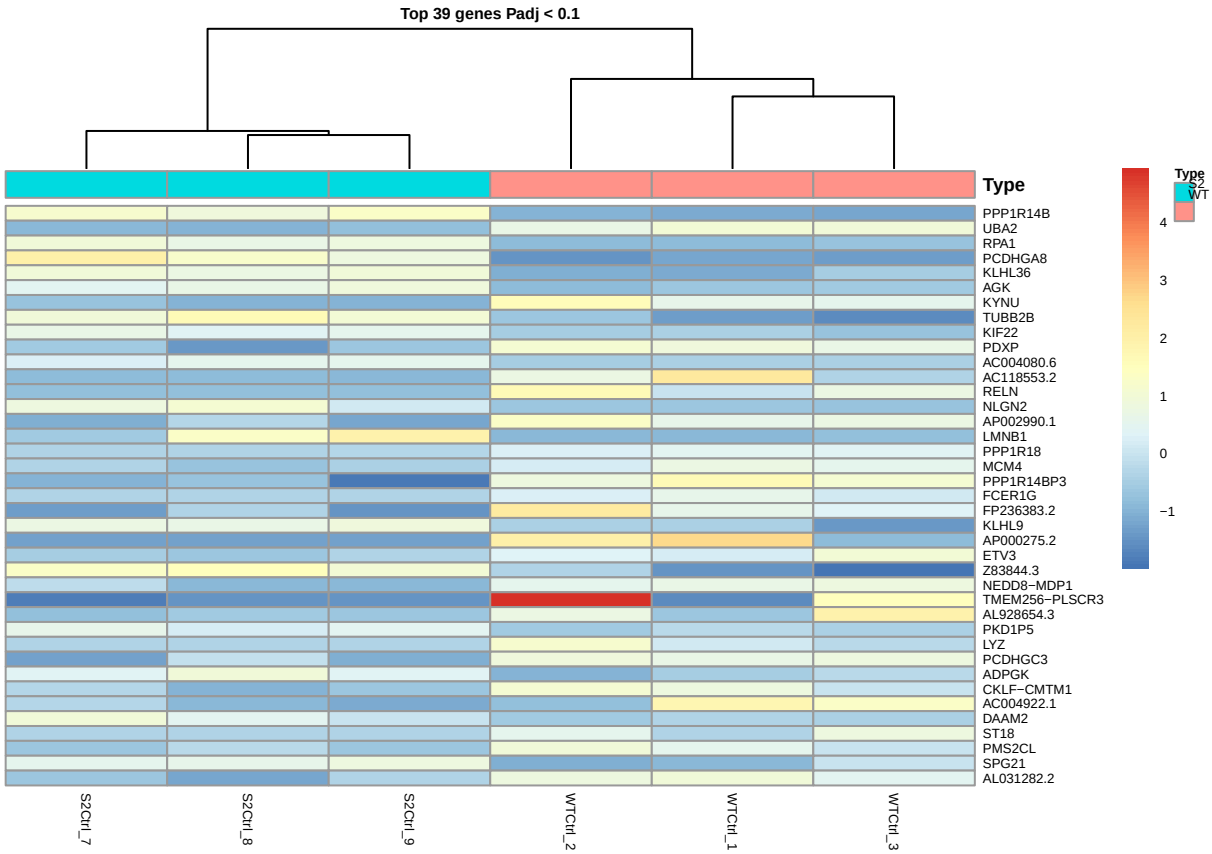
```r
write.csv(pvalue, file="Control_0.05_Pvalue.csv")

padj <- subset(resOrdered, padj < 0.1)

write.csv(padj, file="Control_0.1_Padj.csv")
```

Heatmaps

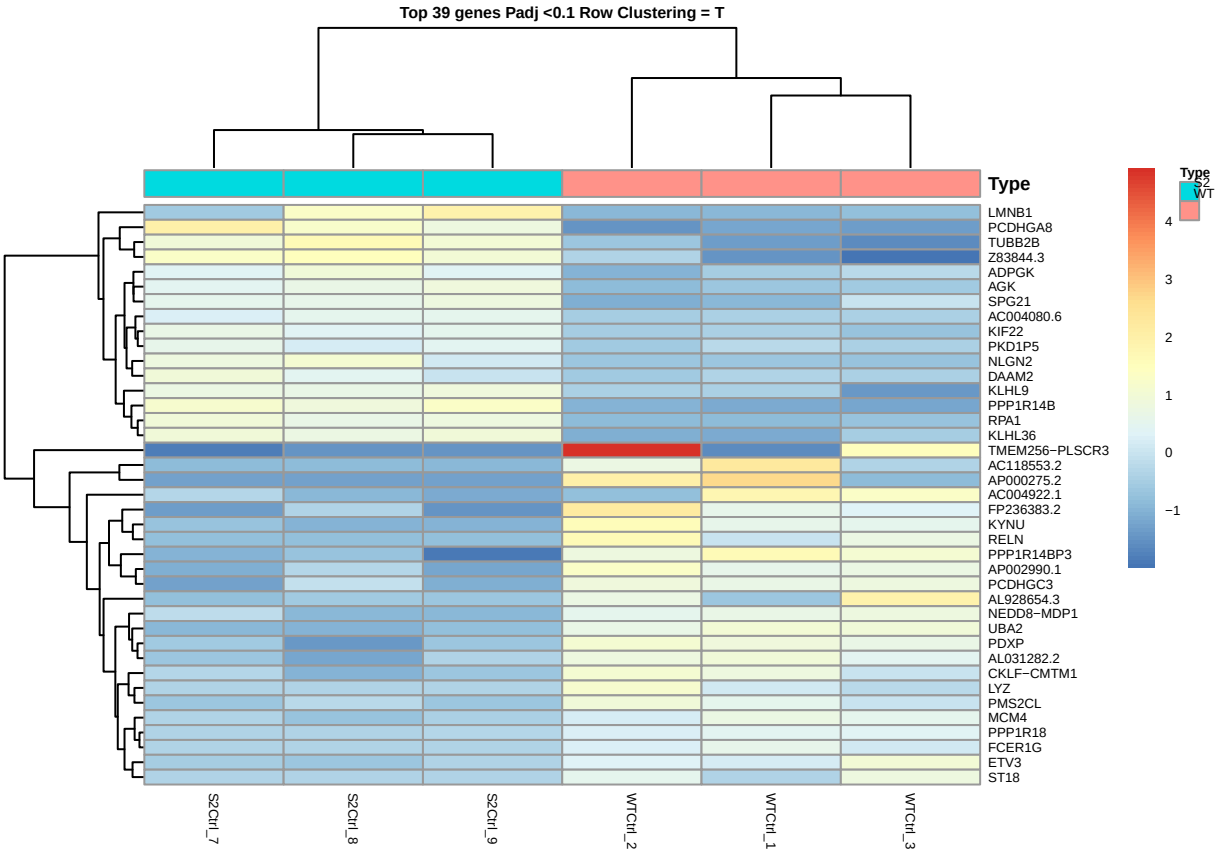
Here I have plotted the top Genes by Pvalue, and Adjusted P value. The rows will correspond to the csv files created in the last step, helping visualize DE across samples. Row orders will not directly correspond to the csv files when clustering = TRUE for rows.

I apologise for the scale, it had to be shrunk to fit in.

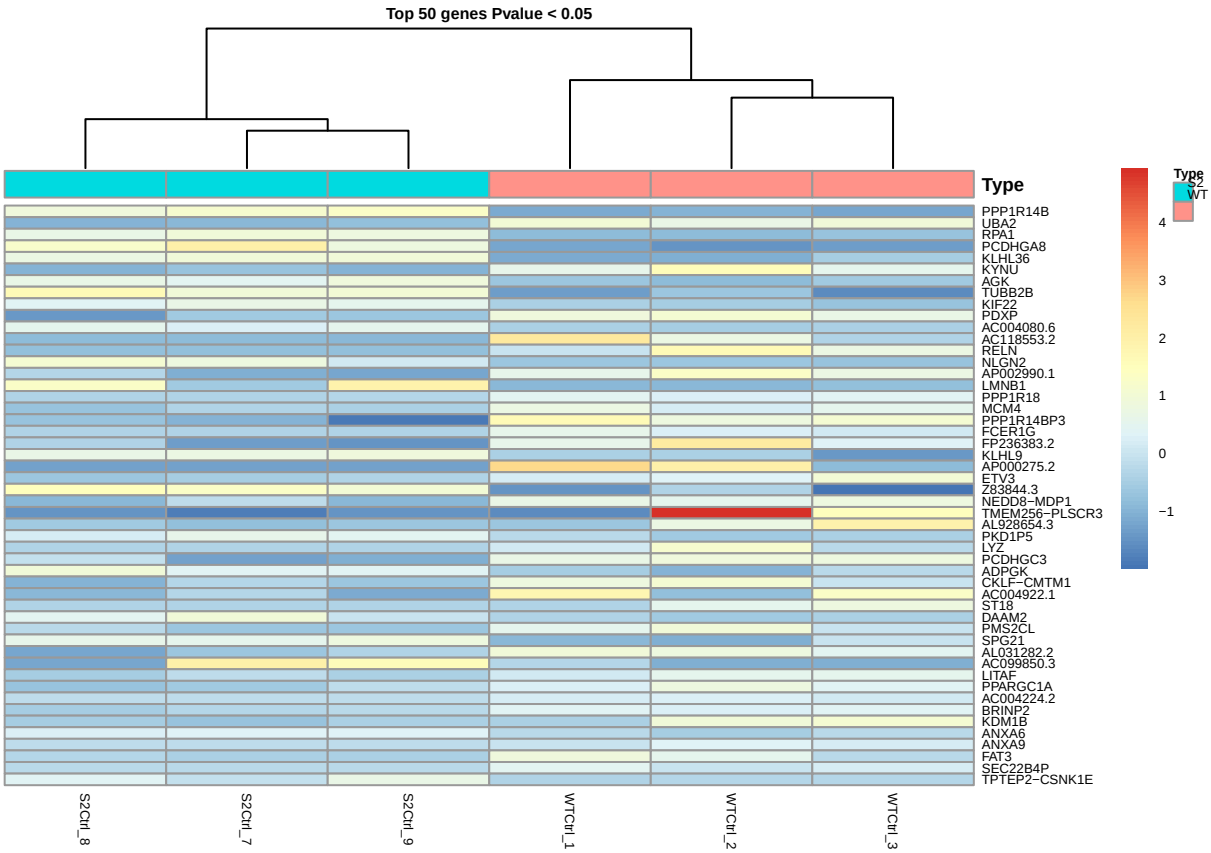
```r
library("pheatmap")
mat = assay(rld)[ head(order(res$padj),39), ] # select the top 30 genes with the lowest padj
mat = mat - rowMeans(mat) # Subtract the row means from each value
# Optional, but to make the plot nicer:
df = as.data.frame(colData(rld)[,c("type")]) # Create a dataframe with a column of the conditions
colnames(df) = "Type" # Rename the column header
rownames(df) = colnames(mat) # add rownames
# and plot the actual heatmap
pheatmap(mat, annotation_col=df, cluster_rows = FALSE, cex=0.7, main = "Top 39 genes Padj < 0.1")
```



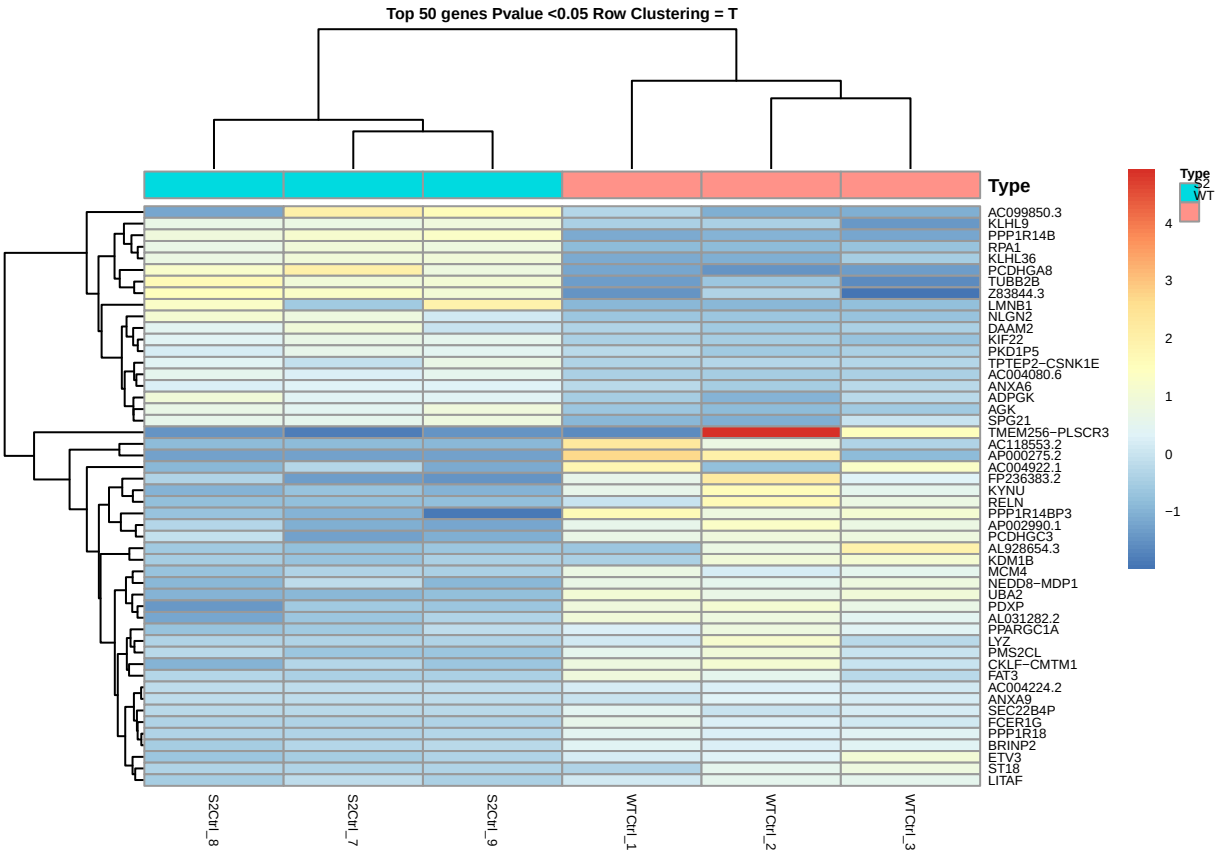
```
mat = assay(rld)[ head(order(res$padj),39), ] # select the top 30 genes with the lowest padj
mat = mat - rowMeans(mat) # Subtract the row means from each value
# Optional, but to make the plot nicer:
df = as.data.frame(colData(rld)[,c("type")]) # Create a dataframe with a column of the conditions
colnames(df) = "Type" # Rename the column header
rownames(df) = colnames(mat) # add rownames
# and plot the actual heatmap
pheatmap(mat, annotation_col=df, cluster_rows = TRUE, cex=0.7, main = "Top 39 genes Padj <0.1 Row Cluste
```




```
library("pheatmap")
mat = assay(rld)[ head(order(res$pvalue),50), ] # select the top 30 genes with the lowest padj
mat = mat - rowMeans(mat) # Subtract the row means from each value
# Optional, but to make the plot nicer:
df = as.data.frame(colData(rld)[,c("type")]) # Create a dataframe with a column of the conditions
colnames(df) = "Type" # Rename the column header
rownames(df) = colnames(mat) # add rownames
# and plot the actual heatmap
pheatmap(mat, annotation_col=df, cluster_rows = FALSE, cex=0.7, main = "Top 50 genes Pvalue < 0.05")
```




```
mat = assay(rld)[ head(order(res$pvalue),50), ] # select the top 30 genes with the lowest padj
mat = mat - rowMeans(mat) # Subtract the row means from each value
# Optional, but to make the plot nicer:
df = as.data.frame(colData(rld)[,c("type")]) # Create a dataframe with a column of the conditions
colnames(df) = "Type" # Rename the column header
rownames(df) = colnames(mat) # add rownames
# and plot the actual heatmap
pheatmap(mat, annotation_col=df, cluster_rows = TRUE, cex=0.7, main = "Top 50 genes Pvalue <0.05 Row Clu
```


Top 50 genes Pvalue <0.05 Row Clustering = T
Type
AC099850.3
KLHL9
PPP1R14B
RPA1
KLHL36
PCDHGA8
TUBB2B
Z83844.3
LMNB1
NLGN2
DAAM2
KIF22
PKD1P5
TPTEP2−CSNK1E
AC004080.6
ANXA6
ADPGK
AGK
SPG21
TMEM256−PLSCR3
AC118553.2
AP000275.2
AC004922.1
FP236383.2
KYNU
RELN
PPP1R14BP3
AP002990.1
PCDHGC3
AL928654.3
KDM1B
MCM4
NEDD8−MDP1
UBA2
PDXP
AL031282.2
PPARGC1A
LYZ
PMS2CL
CKLF−CMTM1
FAT3
AC004224.2
ANXA9
SEC22B4P
FCER1G
PPP1R18
BRINP2
ETV3
ST18
LITAF
S2Ctrl_8
S2Ctrl_7
S2Ctrl_9
WTCtrl_1
WTCtrl_2
WTCtrl_3
Type
S2
WT
4
3
2
1
0
−1