Shank3 Modulates Sleep and Expression of Circadian Transcription Factors differential expression

October 16, 2018

Contents

Description

This is the report of the analysis made for the paper *Shank3 Modulates Sleep and Expression of Circadian Transcription Factors* by *Ashley M. Ingiosi, Taylor Wintler, Hannah Schoch, Kristan G. Singletary, Dario Righelli, Leandro G. Roser, Davide Risso, Marcos G. Frank and Lucia Peixoto.*

Autism Spectrum Disorder (ASD) is the most prevalent neurodevelopmental disorder in the US that often co-presents with sleep problems. Sleep impairments in ASD predict the severity of ASD core diagnostic symptoms and have a considerable impact on the quality of life of caregivers. However, little is known about the underlying molecular mechanism(s) of sleep impairments in ASD. In this study we investigated the role of Shank3, a high confidence ASD gene candidate, in the regulation of sleep. We show that Shank3 mutant mice have problems falling asleep despite accumulating sleep pressure. Using RNA-seq we show that sleep

deprivation doubles the differences in gene expression between mutants and wild types and downregulates circadian transcription factors Per3, Dec2, and Rev-erbα. Shank3 mutants also have trouble regulating locomotor activity in the absence of light. Overall, our study shows that Shank3 is an important modulator of sleep and circadian activity.

Importing data

Importing data and filtering out those genes with cpm lesser than 1. We use the *filtered.data* method in *NOISeq* package.

```
countMatrix <- ReadDataFrameFromTsv(file.name.path="./data/refSEQ_countMatrix.txt")
```

```
## ./data/refSEQ_countMatrix.txt read from disk!
```

```
# head(countMatrix)

designMatrix <- ReadDataFrameFromTsv(file.name.path="./design/all_samples_short_names.tsv")
```

```
## ./design/all_samples_short_names.tsv read from disk!
```

```
# head(designMatrix)

filteredCountsProp <- filterLowCounts(counts.dataframe=countMatrix,
                                      is.normalized=FALSE,
                                      design.dataframe=designMatrix,
                                      cond.col.name="gcondition",
                                      method.type="Proportion")
```

```
## features dimensions before normalization: 27179
```

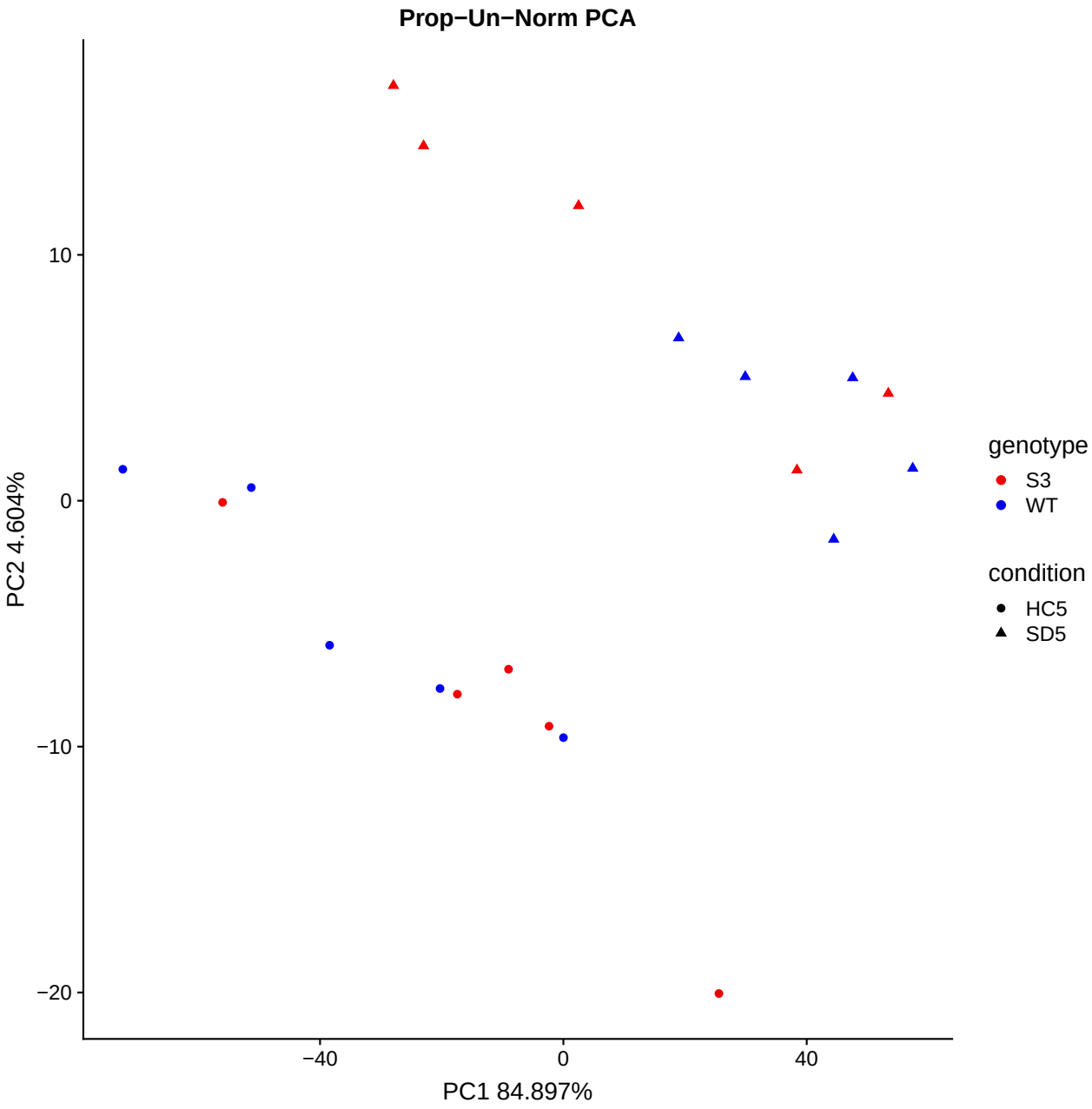
```
## Filtering out low count features...
## 14454 features are to be kept for differential expression analysis with filtering method 3
```

Plot PCA of log unnormalized data

PCA Plot of filtered not-normalized data.

```
PlotPCAPlotlyFunction(counts.data.frame=log1p(filteredCountsProp),
    design.matrix=designMatrix,
    shapeColname="condition", colorColname="genotype", xPCA="PC1", yPCA="PC2",
    plotly.flag=FALSE, show.plot.flag=TRUE, prefix.plot="Prop-Un-Norm")
```

```
## [1] FALSE
```



Control Genes

Negative control genes

Loading Negative Control Genes to normalize data

```
library(readxl)

sd.ctrls <- read_excel(path="./data/controls/Additional File 4 full list of BMC genomics SD&RS2.xlsx", s
sd.ctrls <- sd.ctrls[order(sd.ctrls$adj.P.Val),]

sd.neg.ctrls <- sd.ctrls[sd.ctrls$adj.P.Val > 0.9, ]
```

```
sd.neg.ctrls <- sd.neg.ctrls$`MGI Symbol`
sd.neg.ctrls <- sd.neg.ctrls[-which(is.na(sd.neg.ctrls))]

int.neg.ctrls <- sd.neg.ctrls
int.neg.ctrls <- unique(int.neg.ctrls)
neg.map <- convertGenesViaMouseDb(gene.list=int.neg.ctrls, fromType="SYMBOL",
                                  "ENTREZID")
# sum(is.na(neg.map$ENTREZID))
neg.ctrls.entrez <- as.character(neg.map$ENTREZID)


ind.ctrls <- which(rownames(filteredCountsProp) %in% neg.ctrls.entrez)
counts.neg.ctrls <- filteredCountsProp[ind.ctrls,]
```

positive control genes

Loading Positive Control Genes to detect them during the differential expression step.

```
## sleep deprivation
sd.lit.pos.ctrls <- read_excel("./data/controls/SD_RS_PosControls_final.xlsx",
                          sheet=1)
colnames(sd.lit.pos.ctrls) <- sd.lit.pos.ctrls[1,]
sd.lit.pos.ctrls <- sd.lit.pos.ctrls[-1,]


sd.est.pos.ctrls <- read_excel("./data/controls/SD_RS_PosControls_final.xlsx",
                          sheet=3)

sd.pos.ctrls <- cbind(sd.est.pos.ctrls$`MGI Symbol`, "est")
sd.pos.ctrls <- rbind(sd.pos.ctrls, cbind(sd.lit.pos.ctrls$Gene, "lit"))

sd.pos.ctrls <- sd.pos.ctrls[-which(duplicated(sd.pos.ctrls[,1])),]
sd.pos.ctrls <- sd.pos.ctrls[-which(is.na(sd.pos.ctrls[,1])),]
```

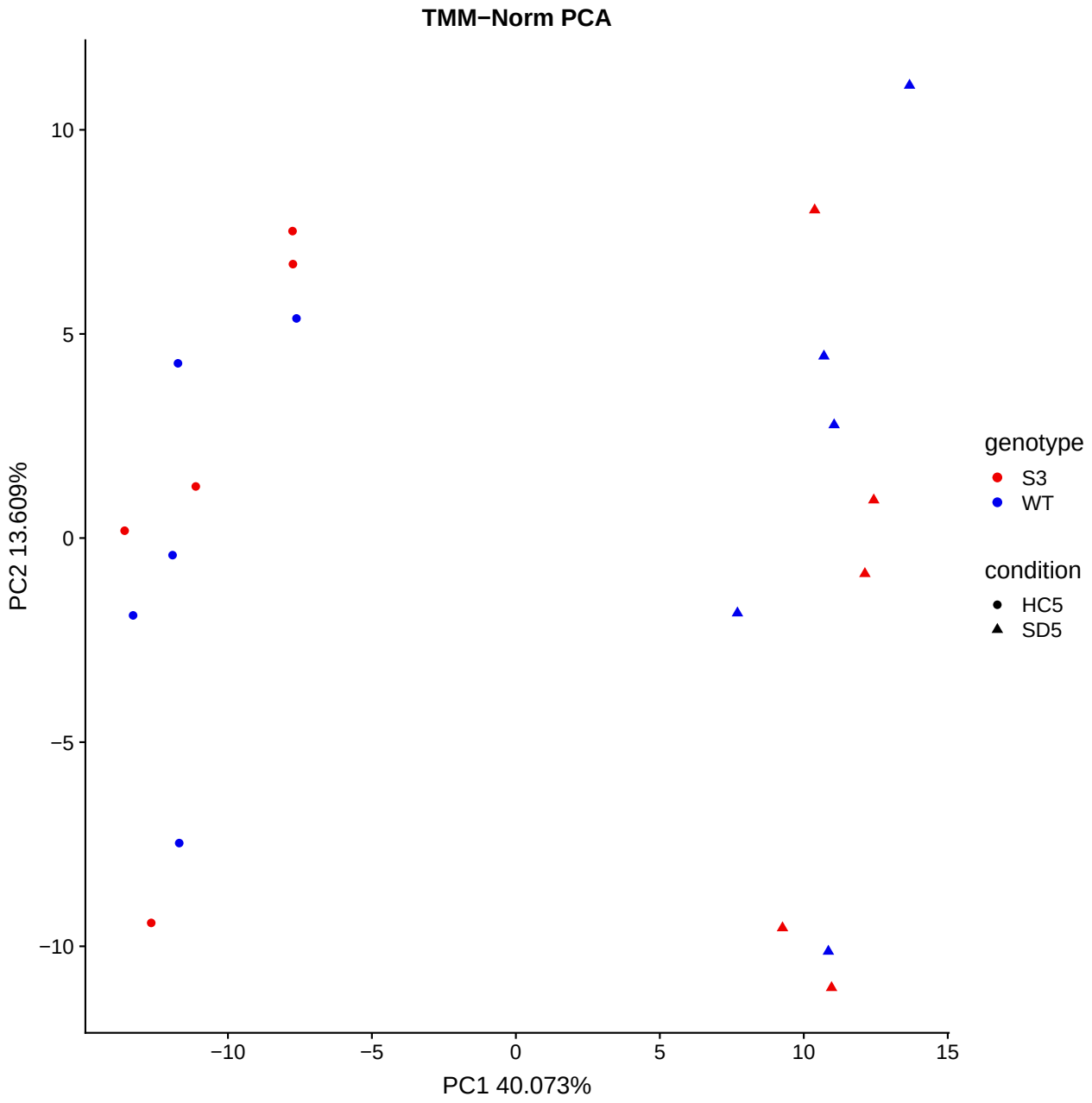
Normalizations

TMM Normalization

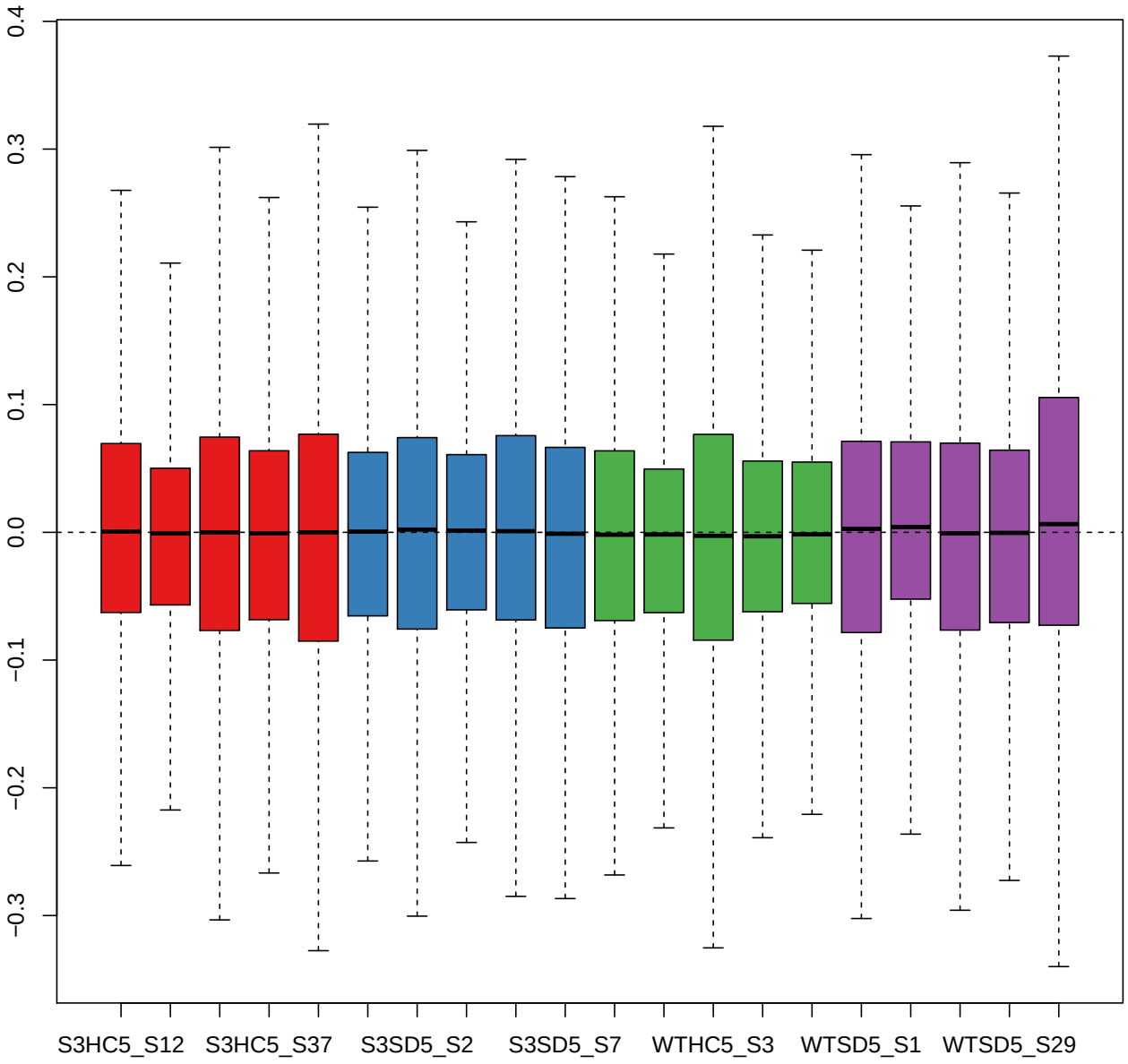
Normalizing data with *TMM*, as implemented in *edgeR* package, and plotting a PCA and an RLE plot of them.

```
normPropCountsUqua <- NormalizeData(data.to.normalize=filteredCountsProp,
                                    norm.type="tmm",
                                    design.matrix=designMatrix)

PlotPCAPlotlyFunction(counts.data.frame=log1p(normPropCountsUqua),
                      design.matrix=designMatrix, shapeColname="condition",
                      colorColname="genotype", xPCA="PC1", yPCA="PC2",
                      plotly.flag=FALSE, show.plot.flag=TRUE,
                      prefix.plot="TMM-Norm")
```

```
## [1] FALSE
```




```
pal <- RColorBrewer::brewer.pal(9, "Set1")
plotRLE(as.matrix(normPropCountsUqua), outline=FALSE, col=pal[designMatrix$gcondition])
```



TMM + RUVs Normalization

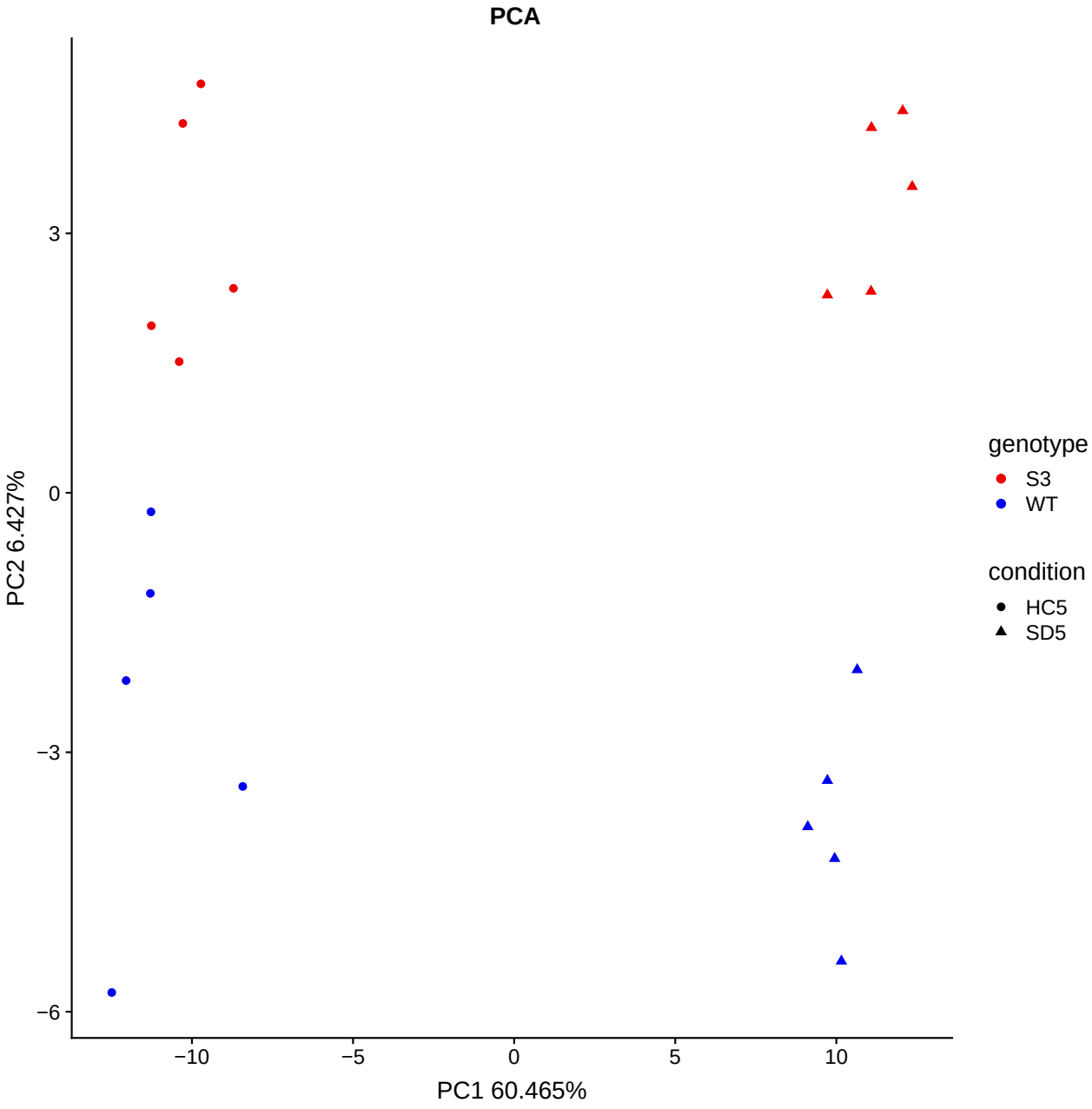
Applying a *RUVs* method of *RUVSeq* package on normalized data, in order to adjust the counts for the unwanted variation. And of corse we plot a PCA and an RLE plot on these data.

```
library(RUVSeq)
neg.ctrl.list <- rownames(counts.neg.ctrls)
groups <- makeGroups(designMatrix$gcondition)
ruvedSExprData <- RUVs(as.matrix(round(normPropCountsUqua)), cIdx=neg.ctrl.list,
                       scIdx=groups, k=5)

normExprData <- ruvedSExprData$normalizedCounts

ggp <- PlotPCAPlotlyFunction(counts.data.frame=log1p(normExprData),
                    design.matrix=designMatrix, shapeColname="condition",
                    colorColname="genotype", xPCA="PC1", yPCA="PC2",
```

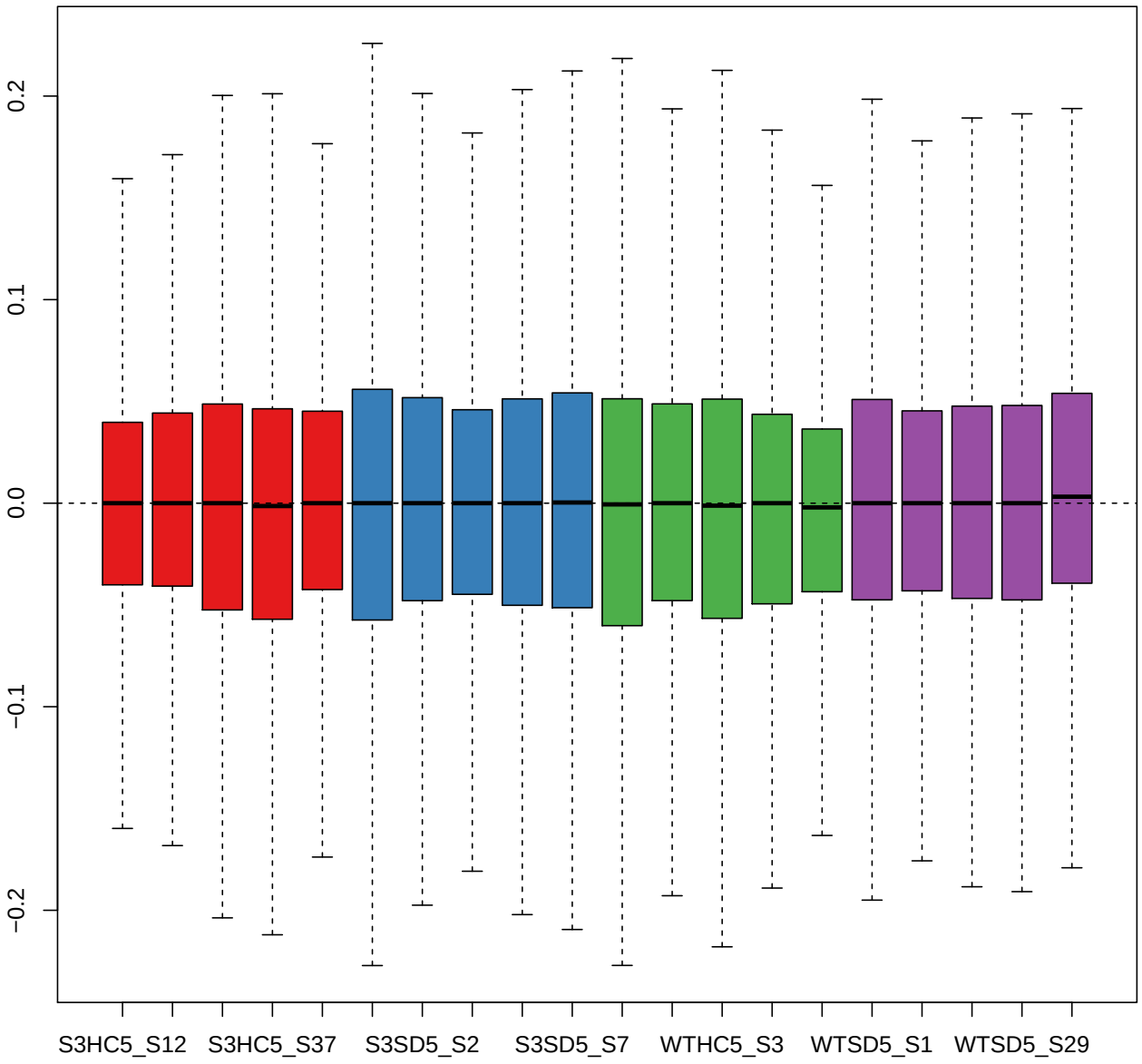
```
                    plotly.flag=FALSE, show.plot.flag=FALSE, save.plot=FALSE,
                    prefix.plot=NULL)
```

```
## [1] FALSE
```

```
print(ggp)
```




```
dir.create("plots")
save_plot(filename="plots/PCA.pdf", plot=ggp)

pal <- RColorBrewer::brewer.pal(9, "Set1")
plotRLE(normExprData, outline=FALSE, col=pal[designMatrix$gcondition])
```



edgeR Differential Expression Analysis

Making differential expression analysis with *edgeR* package on four different contrasts.

Here is a brief legend:

- WTHC5: Wild Type Home Cage Control 5 days
- WTSD5: Wild Type Sleep Deprivation 5 days.
- KOHC5: Knock Out Home Cage Control 5 days.
- KOSD5: Knock Out Sleep Deprivation 5 days.

```
padj.thr <- 0.05
venn.padgj.thr <- 0.1
desMat <- cbind(designMatrix, ruvedSExprData$W)
colnames(desMat) <- c(colnames(designMatrix), colnames(ruvedSExprData$W))
```

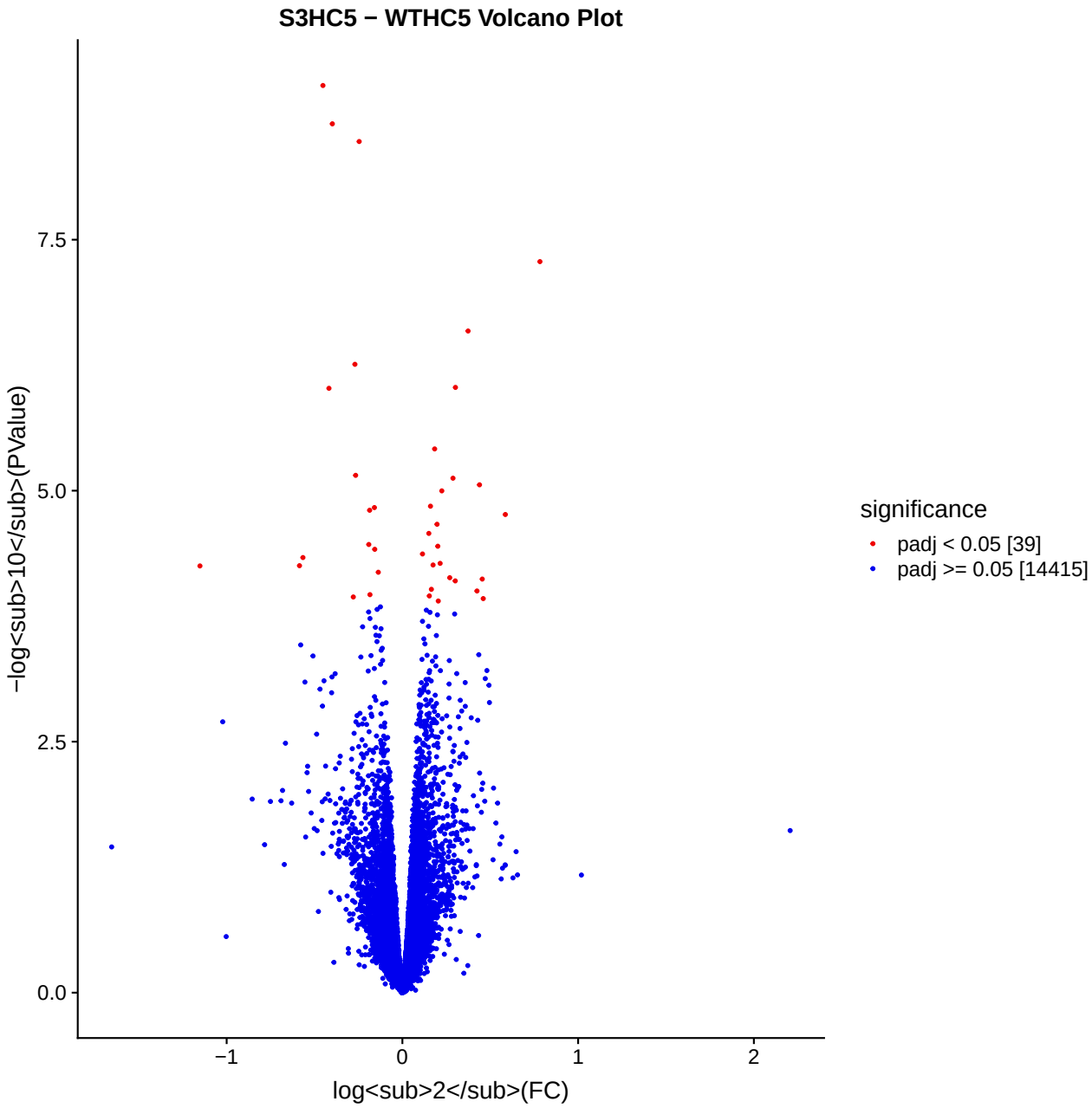
```
cc <- c("S3HC5 - WTHC5", "S3SD5 - WTSD5")

rescList1 <- applyEdgeR(counts=filteredCountsProp, design.matrix=desMat,
                        factors.column="gcondition",
                        weight.columns=c("W_1", "W_2", "W_3", "W_4", "W_5"),
                        contrasts=cc, useIntercept=FALSE, p.threshold=1,
                        is.normalized=FALSE, verbose=TRUE)
names <- names(rescList1)
rescList1 <- lapply(seq_along(rescList1), function(i)
{
    attachMeans(normalized.counts=normExprData, design.matrix=desMat,
                factor.column="gcondition", contrast.name=names(rescList1)[i],
                de.results=rescList1[[i]])
})
names(rescList1) <- names
```

Shank3 Home Cage control VS Wild Type Home Cage Controls

volcano plot

A volcano plot of differential expressed genes.

```
res.o.map1 <- convertGenesViaMouseDb(gene.list=rownames(rescList1[[1]]),
                             fromType="ENTREZID")

res.o <- attachGeneColumnToDf(mainDf=rescList1[[1]],
                                genesMap=res.o.map1,
                                rowNamesIdentifier="ENTREZID",
                                mapFromIdentifier="ENTREZID",
                                mapToIdentifier="SYMBOL")
WriteDataFrameAsTsv(data.frame.to.save=res.o,
                    file.name.path=paste0(names(rescList1)[1], "_edgeR"))

vp <- luciaVolcanoPlot(res.o, prefix=names(rescList1)[1],
                        positive.controls.df=NULL,
                        threshold=padj.thr)
print(vp)
```




```
de <- sum(res.o$FDR < padj.thr)
nde <- sum(res.o$FDR >= padj.thr)
detable <- cbind(de,nde)
rownames(detable) <- names(rescList1)[1]
ddetable <- detable

tot.ctrls <- dim(sd.pos.ctrls)[1]
idx.pc <- which(tolower(res.o$gene) %in% tolower(sd.pos.ctrls[,1]))
tot.pc.de <- sum(res.o$FDR[idx.pc] < padj.thr)
tot.pc.nde <- length(idx.pc) - tot.pc.de

wt <- res.o[which(res.o$FDR < padj.thr),]
wt.sign.genes.entrez <- rownames(res.o)[which(res.o$FDR < venn.padgj.thr)]
```

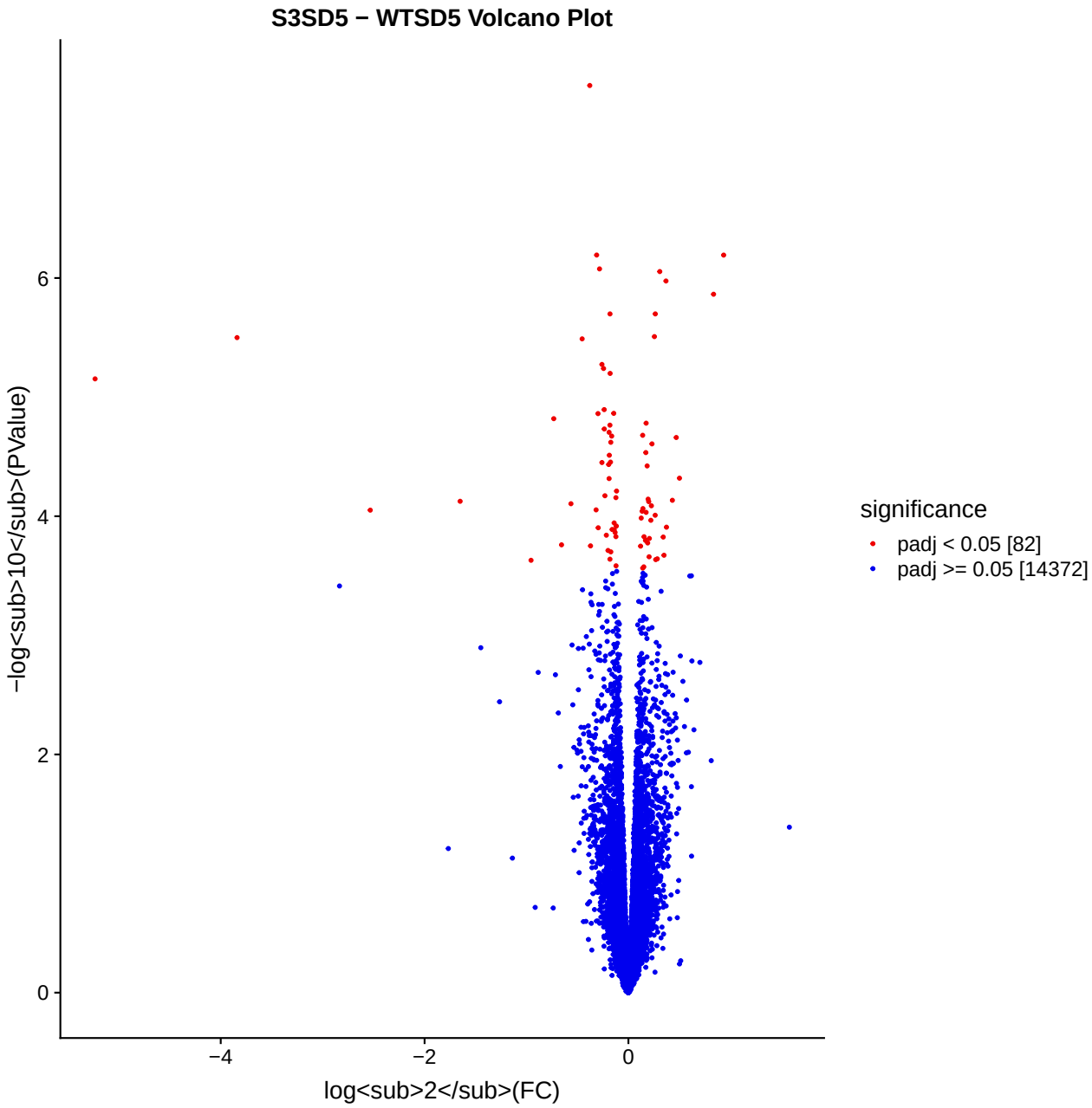
```
kowthc5 <- res.o[which(res.o$FDR < padj.thr),]
kowthc5.sign.genes.entrez <- rownames(res.o)[which(res.o$FDR < venn.padgj.thr)]
```

Shank3 Sleed Deprivation VS Wild Type Sleep Deprivation

volcano plot

A volcano plot of differential expressed genes.

```
rs2.o.map <- convertGenesViaMouseDb(gene.list=rownames(rescList1[[2]]),
                            fromType="ENTREZID")

res.rs2.o <- attachGeneColumnToDf(mainDf=rescList1[[2]],
                                  genesMap=rs2.o.map,
                                  rowNamesIdentifier="ENTREZID",
                                  mapFromIdentifier="ENTREZID",
                                  mapToIdentifier="SYMBOL")
WriteDataFrameAsTsv(data.frame.to.save=res.rs2.o,
                    file.name.path=paste0(names(rescList1)[2], "_edgeR"))

vp <- luciaVolcanoPlot(res.rs2.o, positive.controls.df=NULL,
                       prefix=names(rescList1)[2],
                       threshold=padj.thr)
print(vp)
```




```
de <- sum(res.rs2.o$FDR < padj.thr)
nde <- sum(res.rs2.o$FDR >= padj.thr)
detable <- cbind(de,nde)
rownames(detable) <- names(rescList1)[2]
ddetable <- rbind(ddetable, detable)
pos.df <- cbind(tot.ctrls, tot.pc.de, tot.pc.nde)
colnames(pos.df) <- c("total_p.ctrl", "p.ctrl_de_mapped",
                      "p.ctrl_notde_mapped")
rownames(pos.df) <- names(rescList1)[2]

kowtsd5 <- res.rs2.o[which(res.rs2.o$FDR < padj.thr),]
kowtsd5.sign.genes.entrez <- rownames(res.rs2.o)[which(res.rs2.o$FDR < venn.padgj.thr)]
```

DE TABLE + Positive Controls table

We present a summarization of the results. The first table is a summarization on how many genes are Differentially Expressed. The second table explains on the first column how many positive controls we have, on the second column how many positive controls have been identified over the differentially expressed genes, and, finally, on the third column how many positive controls have beed identified on the NOT differentially expressed genes.

```
ddetable
```

```
##                de   nde
## S3HC5 - WTHC5 39 14415
## S3SD5 - WTSD5 82 14372
```

```
pos.df
```

```
##                total_p.ctrl p.ctrl_de_mapped p.ctrl_notde_mapped
## S3SD5 - WTSD5           579                3                 553
```

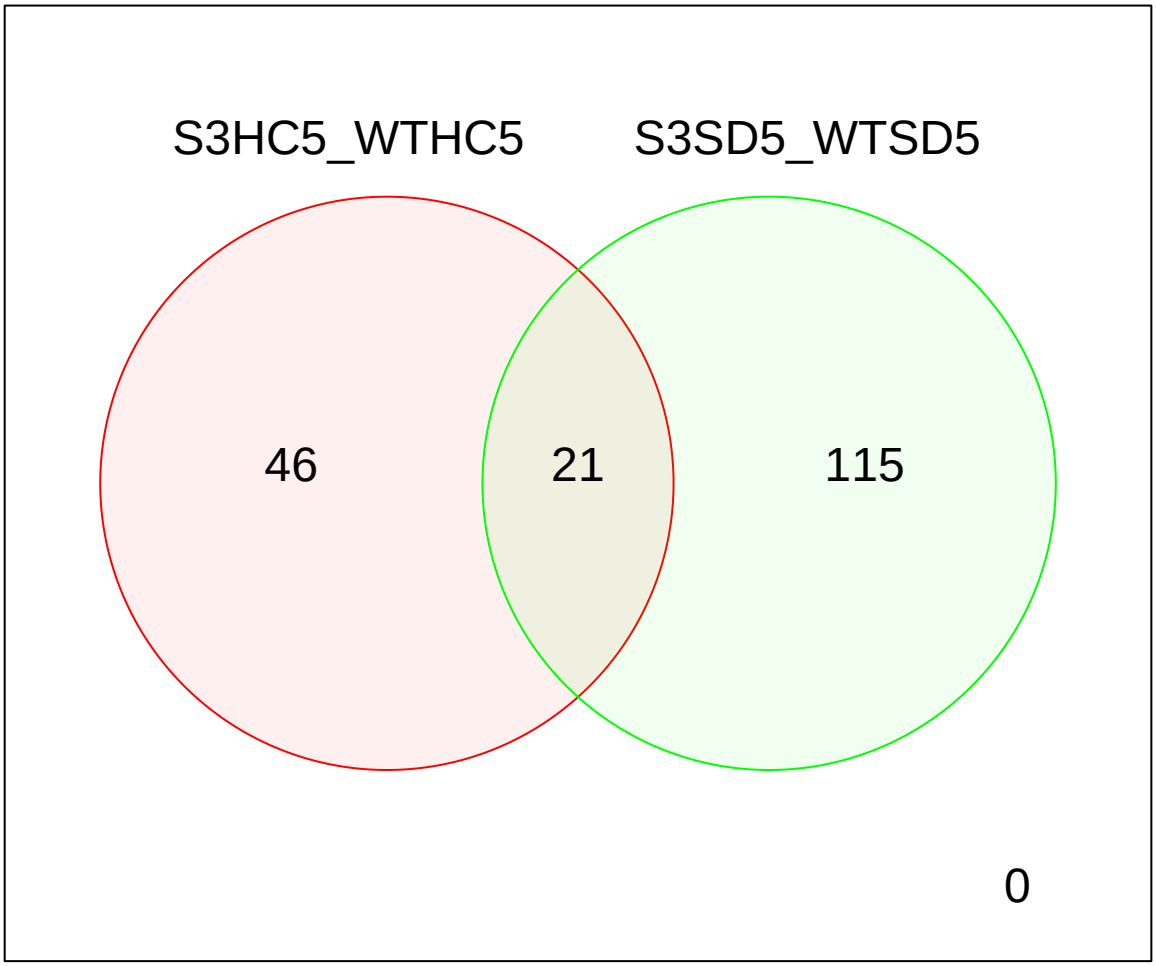
Venn Diagram

KOHC5-WTHC5 vs KOSD5-WTSD5

We take the results of the two contrasts. *Knock Out Sleed Deprivation VS Wild Type Sleep Deprivation* and *Knock Out Home Cage control VS Wild Type Home Cage Controls* . And plot the results in a Venn Diagram

```
source("./R/venn2.R")

gene.map <- convertGenesViaMouseDb(gene.list=rownames(normExprData),
                                   fromType="ENTREZID", toType="SYMBOL")
venn <- Venn2de(x=kowthc5.sign.genes.entrez, y=kowtsd5.sign.genes.entrez,
        label1="S3HC5_WTHC5", label2="S3SD5_WTSD5",
        title="S3HC5_WTHC5 venn S3SD5_WTSD5", plot.dir="./",
        conversion.map=gene.map)
```

S3HC5_WTHC5 venn S3SD5_WTSD5



Heatmaps

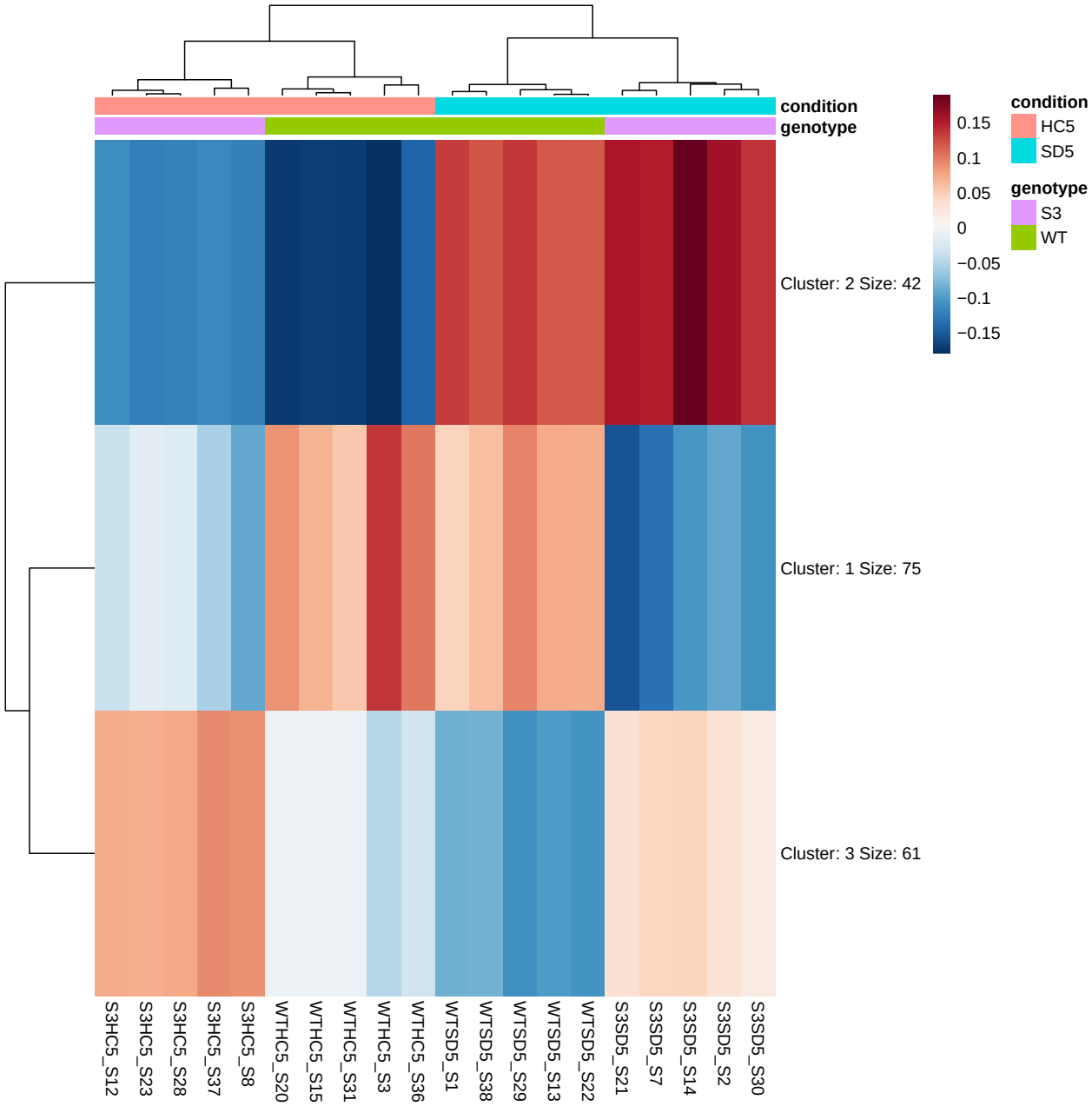
Setting up the data structures for the heatmps.

```
source("./R/heatmapFunctions.R")
de.genes.entr <- union(rownames(venn$int), rownames(venn$XnoY))
de.genes.entr <- union(de.genes.entr, rownames(venn$YnoX))

gene.map <- convertGenesViaMouseDb(gene.list=de.genes.entr,
                            fromType="ENTREZID")
```

```
de.genes.symb <- attachGeneColumnToDf(as.data.frame(de.genes.entr,
                                                    row.names=de.genes.entr),
                                      genesMap=gene.map,
                                      rowNamesIdentifier="ENTREZID",
                                      mapFromIdentifier="ENTREZID",
                                      mapToIdentifier="SYMBOL")

# de.genes.symb[which(is.na(de.genes.symb$gene)),]
de.genes.symb$gene[which(de.genes.symb$de.genes.entr=="100039826")]  <- "Gm2444" ## not annotated in nc
de.genes.symb$gene[which(de.genes.symb$de.genes.entr=="210541")]  <- "Entrez:210541" ## not annotated i


de.genes.counts <- normExprData[match(de.genes.symb$de.genes.entr, rownames(normExprData)),]
rownames(de.genes.counts) <- de.genes.symb$gene

de.gene.means <- computeGeneMeansOverGroups(counts=de.genes.counts,
                            design=designMatrix, groupColumn="gcondition")

library(gplots)
library(clusterExperiment)
color.palette = clusterExperiment::seqPal3#c("black", "yellow")
pal <- colorRampPalette(color.palette)(n = 1000)


library(pheatmap)
filter2 <- rowMeans(de.gene.means)>0
filter <- apply(de.gene.means, 1, function(x) log(x[4]/x[3]) * log(x[2]/x[1]) < 0)
filter[is.na(filter)] <- FALSE
```

Heatmap gene by bene

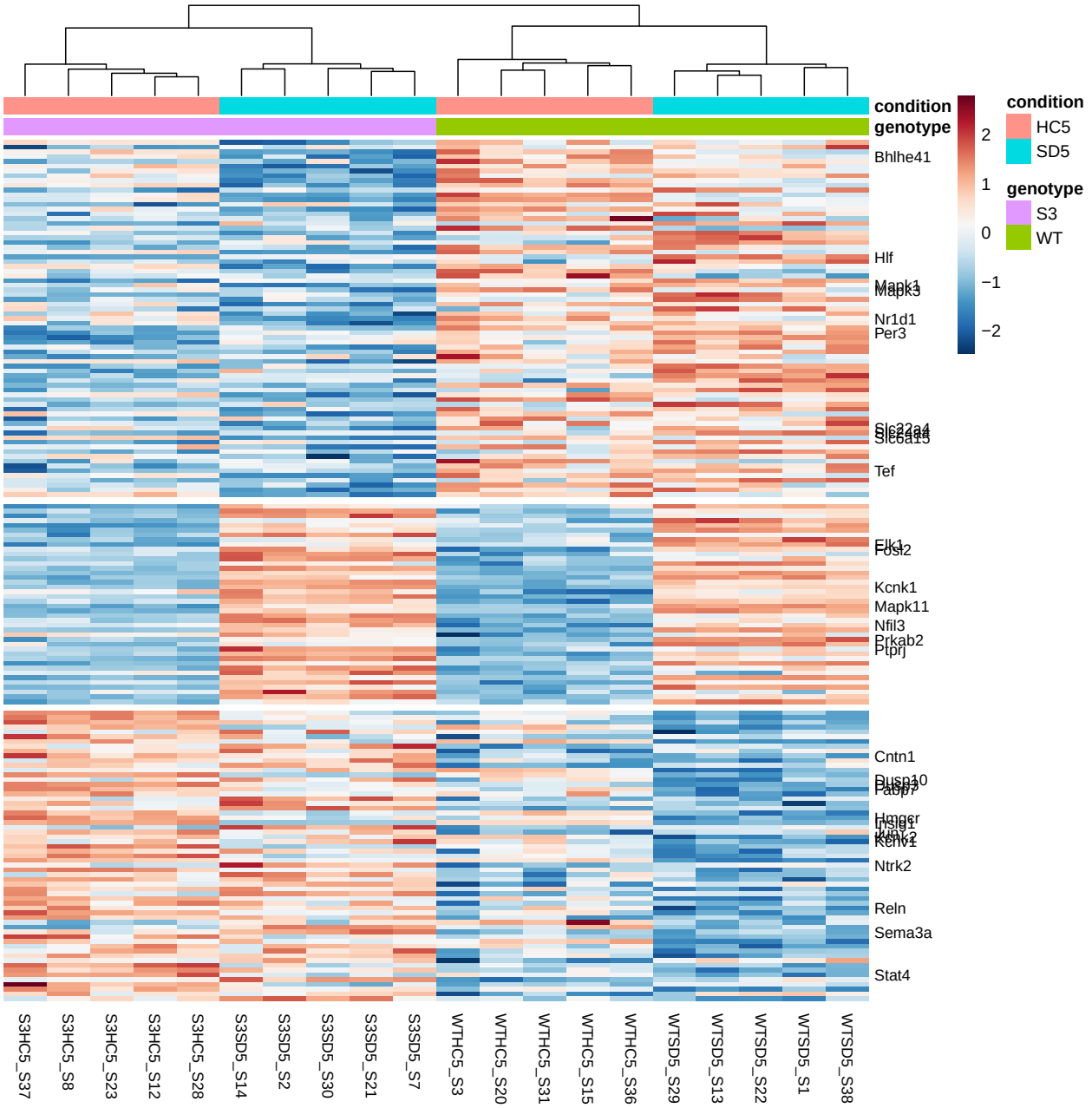
```
gene_names <- c("Nr1d1", "Fabp7", "Per3", "Jun", "Elk1", "Fosl2", "Mapk1",
                "Mapk3", "Mapk11", "Hmgcr", "Insig1", "Nfil3", "Stat4",
                "Kcnv1", "Kcnk1", "Kcnk2", "Dusp10", "Dusp3", "Ptprj",
                "Cntn1", "Ntrk2", "Reln", "Sema3a", "Tef", "Hlf", "Nr1d1",
                "Prkab2", "Bhlhe41", "Slc6a15", "Slc22a4", "Slc24a4")
idx <- which(!(rownames(de.genes.counts) %in% gene_names))
de.genes.counts1 <- de.genes.counts
rownames(de.genes.counts1)[idx] <- ""
ann.col <- desMat[, c(1:2)]
de.heatmap <- de.genes.counts[filter2,]
set.seed(0)
heatmap_data <- t(scale(t(log(de.heatmap+1)), center = TRUE, scale = FALSE))
ph1 <- pheatmap(heatmap_data, cluster_cols=TRUE, scale="none",
            color=pal, border_color=NA, fontsize_row=10, kmeans_k=3, annotation_col=ann.col)
```




```
clusterized.genes <- as.data.frame(ph1$kmeans$cluster)

gene.map <- convertGenesViaMouseDb(gene.list=rownames(clusterized.genes), fromType="SYMBOL")
converted.clusterized.gens <- attachGeneColumnToDf(mainDf=clusterized.genes, genesMap=gene.map,
                  rowNamesIdentifier="SYMBOL", mapFromIdentifier="SYMBOL", mapToIdentifier="ENTREZID")

converted.clusterized.gens$gene[which(rownames(converted.clusterized.gens)=="Gm2444")]  <- "100039826" #
converted.clusterized.gens$gene[which(rownames(converted.clusterized.gens)=="Entrez:210541")]  <- "2105
converted.clusterized.gens <- converted.clusterized.gens[order(converted.clusterized.gens$`ph1$kmeans$cl

save_pheatmap_pdf(filename="plots/heatmap_kmeans_k3.pdf", plot=ph1, width=20, height=20)
```

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

```
WriteDataFrameAsTsv(data.frame.to.save=converted.clusterized.gens, file.name.path="plots/clustered_genes

ord.de.genes.counts <- de.heatmap[match(rownames(converted.clusterized.gens), rownames(de.heatmap)),]
idx <- which(!(rownames(ord.de.genes.counts) %in% gene_names))

rownames(ord.de.genes.counts)[idx] <- ""
gaps.row <- c()
for(i in c(1:3))
{
    li <- length(which(converted.clusterized.gens$`ph1$kmeans$cluster`==i))
    l <- ifelse(i!=1, gaps.row[i-1]+li, li)
    gaps.row <- c(gaps.row, l)
}

heatmap_data_scaled <- t(scale(t(log(ord.de.genes.counts+1)), center = TRUE, scale = TRUE))

library(dendextend)
column_dend <- as.dendrogram(hclust(dist(t(heatmap_data_scaled))))
ord <- labels(column_dend)
ord[11:15] <- labels(column_dend)[16:20]
ord[16:20] <- labels(column_dend)[11:15]
column_dend <- rotate(column_dend, ord)

ph1 <- pheatmap(heatmap_data_scaled, cluster_cols=as.hclust(column_dend), scale="none",
            color=pal, border_color=NA, fontsize_row=9, fontsize_col=9, cluster_rows=FALSE,
            annotation_col=ann.col, gaps_row=gaps.row)
```



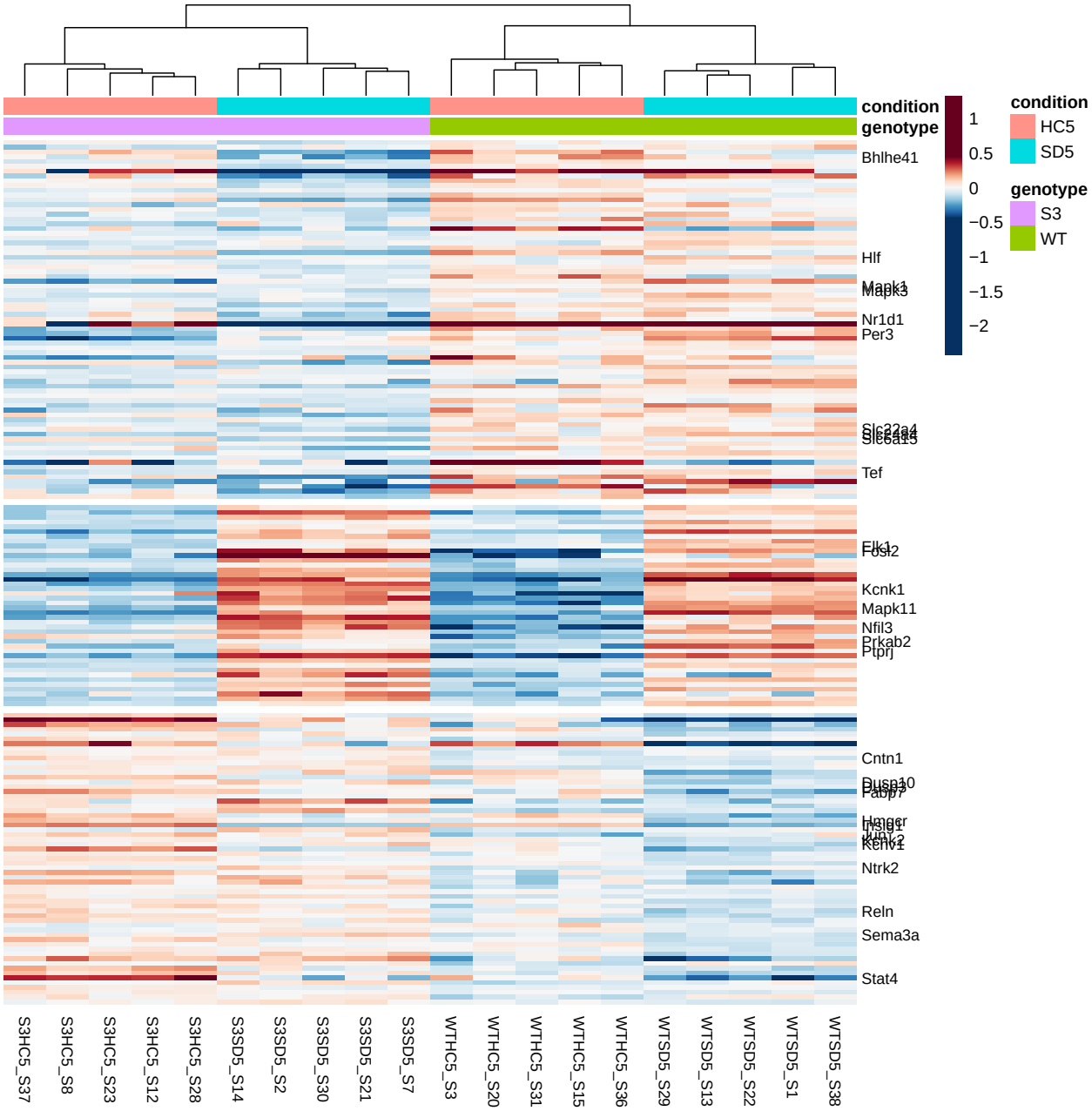

```
save_pheatmap_pdf(filename="plots/heatmap_gg_k3.pdf", plot=ph1, width=20, height=20)
```

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

other heatmaps

```
heatmap_data <- t(scale(t(log(ord.de.genes.counts+1)), center = TRUE, scale = FALSE))

ph1 <- pheatmap(heatmap_data, cluster_cols=as.hclust(column_dend), scale="none",
            color=pal, border_color=NA, fontsize_row=9, fontsize_col=9, cluster_rows=FALSE,
            annotation_col=ann.col, gaps_row=gaps.row,
```

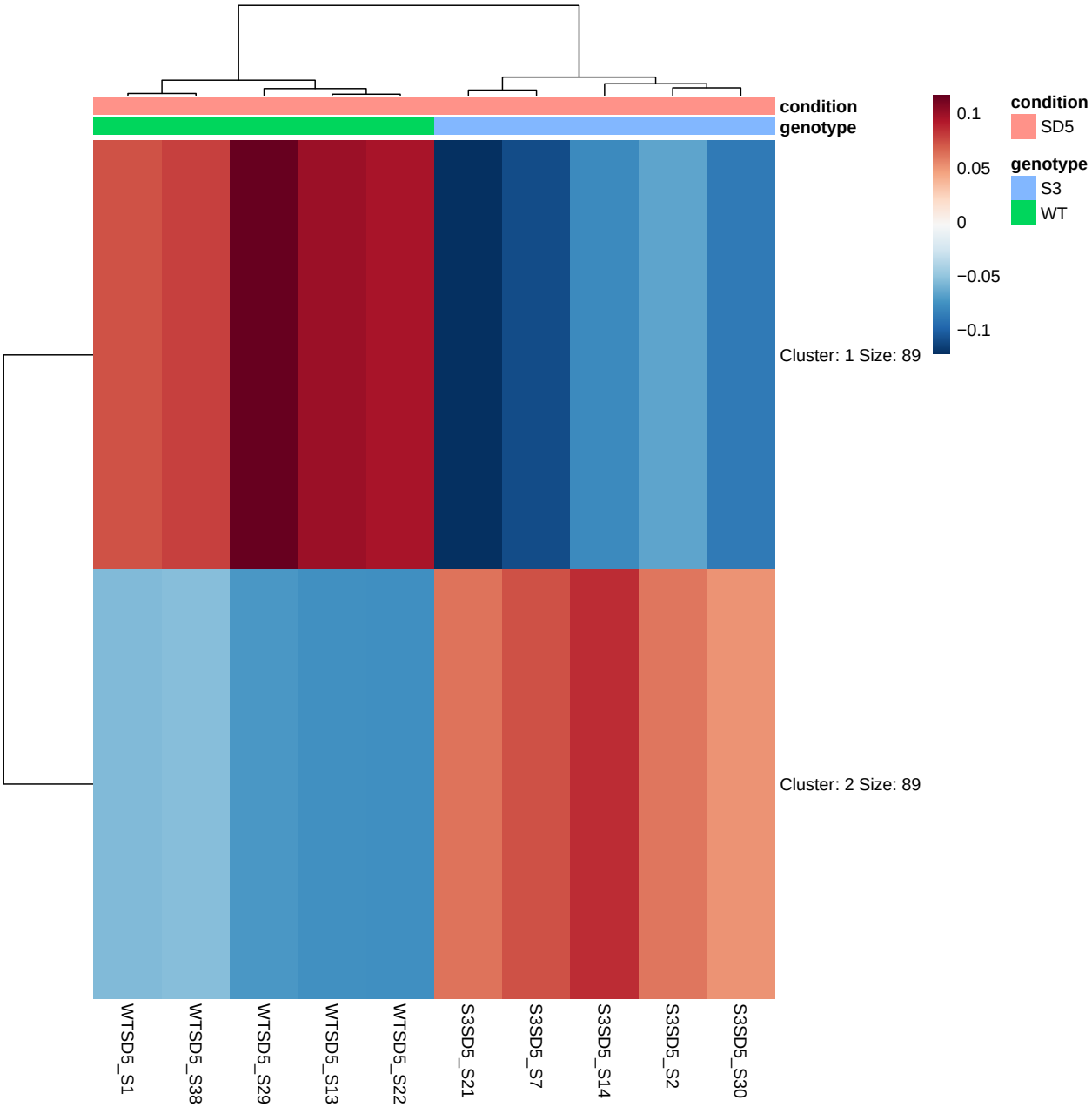
```
            breaks = c(min(heatmap_data), seq(quantile(as.vector(heatmap_data), .01), quantile(as.vecto
```




```
save_pheatmap_pdf(filename="plots/heatmap_gg_k3_no_scale.pdf", plot=ph1, width=20, height=20)
```

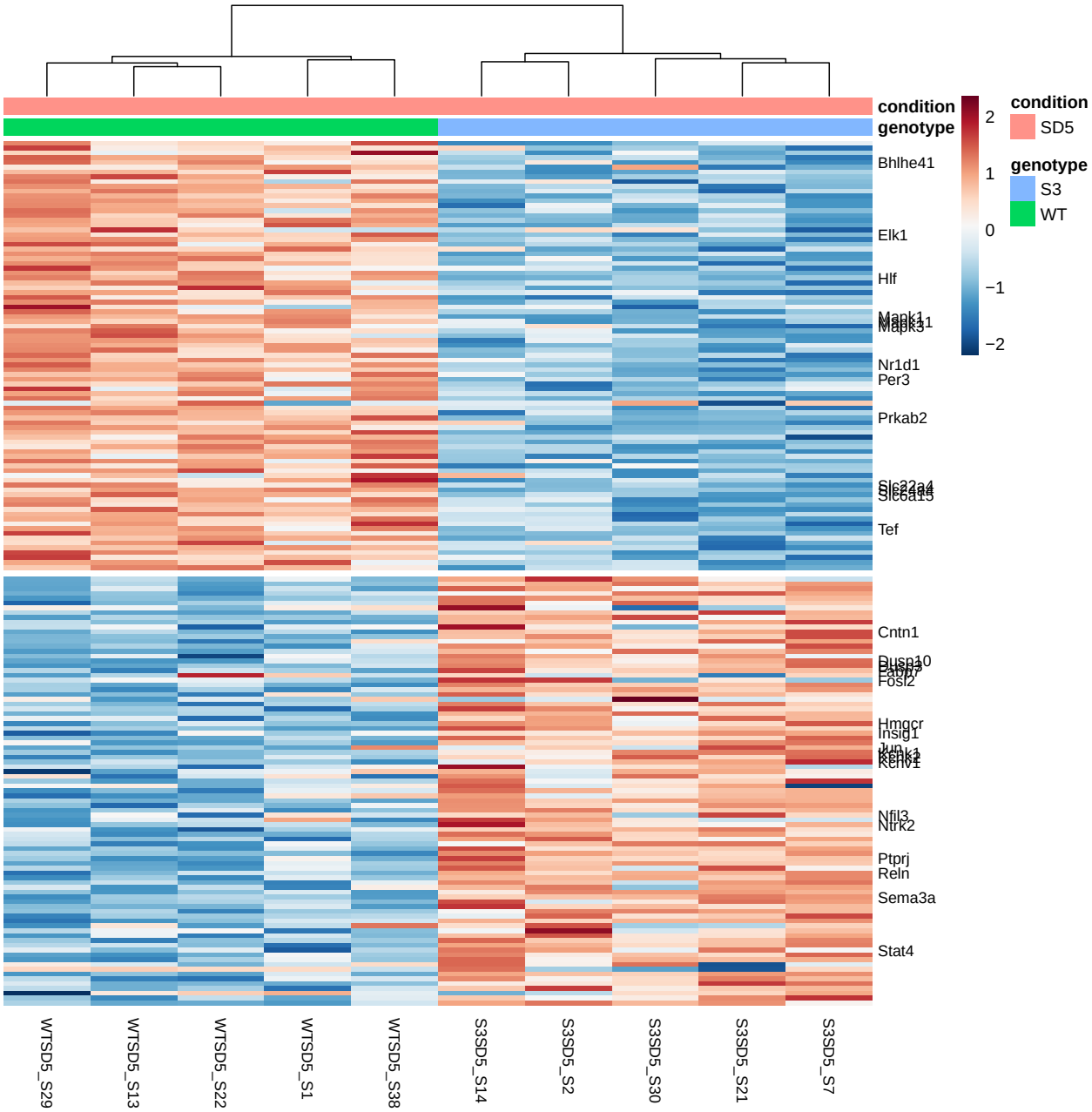
```
## pdf
##   2
```

```
## Only SD samples
heatmap_data <- t(scale(t(log(de.heatmap[, grep("^SD", desMat[,2])]+1)), center = TRUE, scale = FALSE))
ph1 <- pheatmap(heatmap_data, cluster_cols=TRUE, scale="none",
            color=pal, border_color=NA, fontsize_row=10, kmeans_k=2, annotation_col=ann.col)
```




```
clusterized.genes <- as.data.frame(ph1$kmeans$cluster)

gene.map <- convertGenesViaMouseDb(gene.list=rownames(clusterized.genes), fromType="SYMBOL")
converted.clusterized.gens <- attachGeneColumnToDf(mainDf=clusterized.genes, genesMap=gene.map,
                    rowNamesIdentifier="SYMBOL", mapFromIdentifier="SYMBOL", mapToIdentifier="ENTREZID")

converted.clusterized.gens$gene[which(rownames(converted.clusterized.gens)=="Gm2444")]  <- "100039826" #
converted.clusterized.gens$gene[which(rownames(converted.clusterized.gens)=="Entrez:210541")]  <- "2105
converted.clusterized.gens <- converted.clusterized.gens[order(converted.clusterized.gens$`ph1$kmeans$cl

ord.de.genes.counts <- de.heatmap[match(rownames(converted.clusterized.gens), rownames(de.heatmap)),]
idx <- which(!(rownames(ord.de.genes.counts) %in% gene_names))
```

```
rownames(ord.de.genes.counts)[idx] <- ""
gaps.row <- c()
for(i in c(1:2))
{
    li <- length(which(converted.clusterized.gens$`ph1$kmeans$cluster`==i))
    l <- ifelse(i!=1, gaps.row[i-1]+li, li)
    gaps.row <- c(gaps.row, l)
}

heatmap_data <- t(scale(t(log(ord.de.genes.counts[, grep("^SD", desMat[,2])]+1)), center = TRUE, scale =

ph1 <- pheatmap(heatmap_data, cluster_cols=TRUE, scale="none",
            color=pal, border_color=NA, fontsize_row=9, fontsize_col=9, cluster_rows=FALSE,
            annotation_col=ann.col, gaps_row=gaps.row)
```




```
save_pheatmap_pdf(filename="plots/heatmap_gg_sd_only.pdf", plot=ph1, width=20, height=20)
```

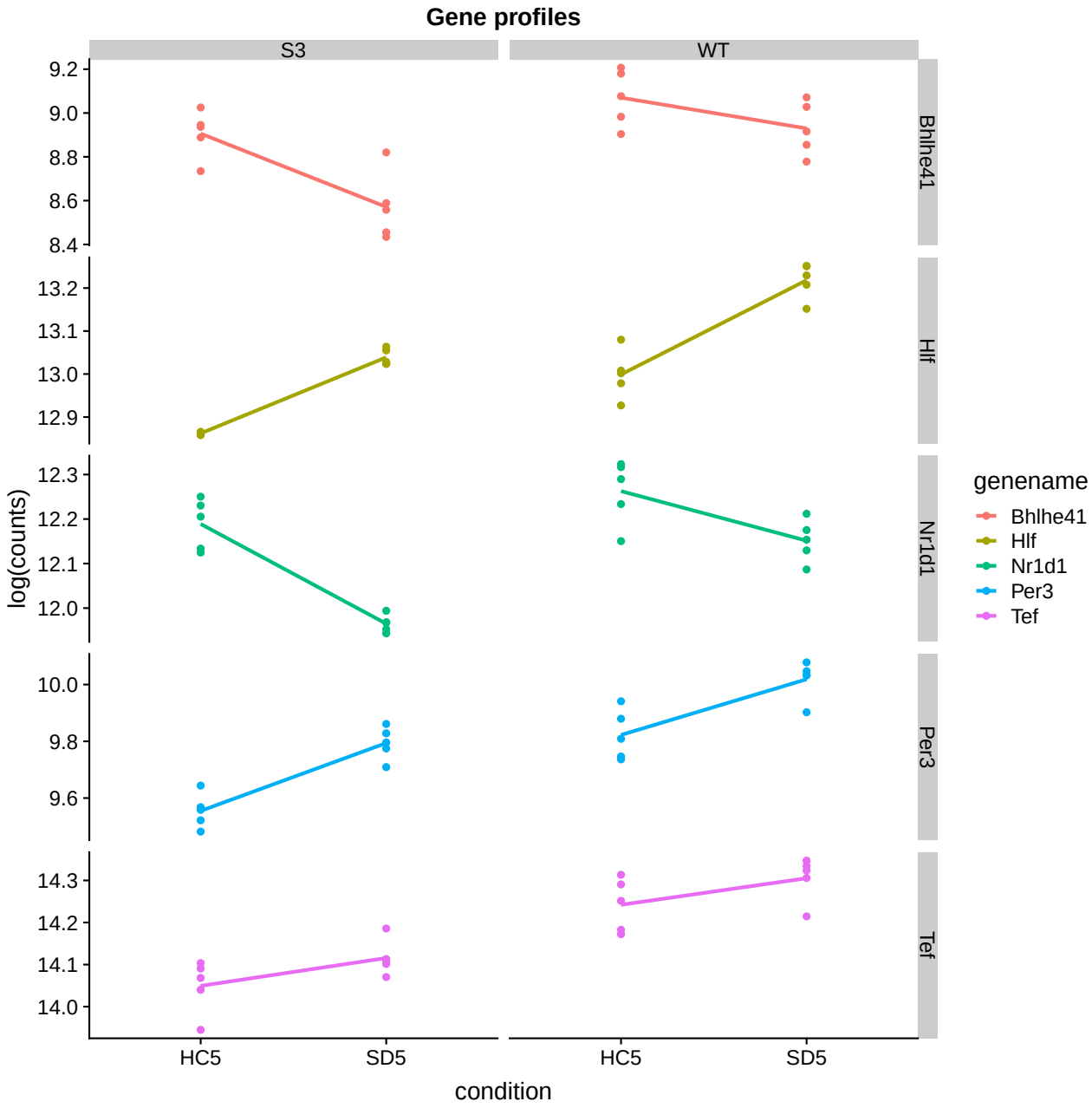
```
## pdf
##   2
```

Group gene profiles

Group gene profiles by genotype

```
g <- geneGroupProfileRows(normalized.counts=normExprData, design.matrix=designMatrix,
           gene.names=c("Nr1d1", "Hlf", "Per3", "Bhlhe41", "Tef"),
```

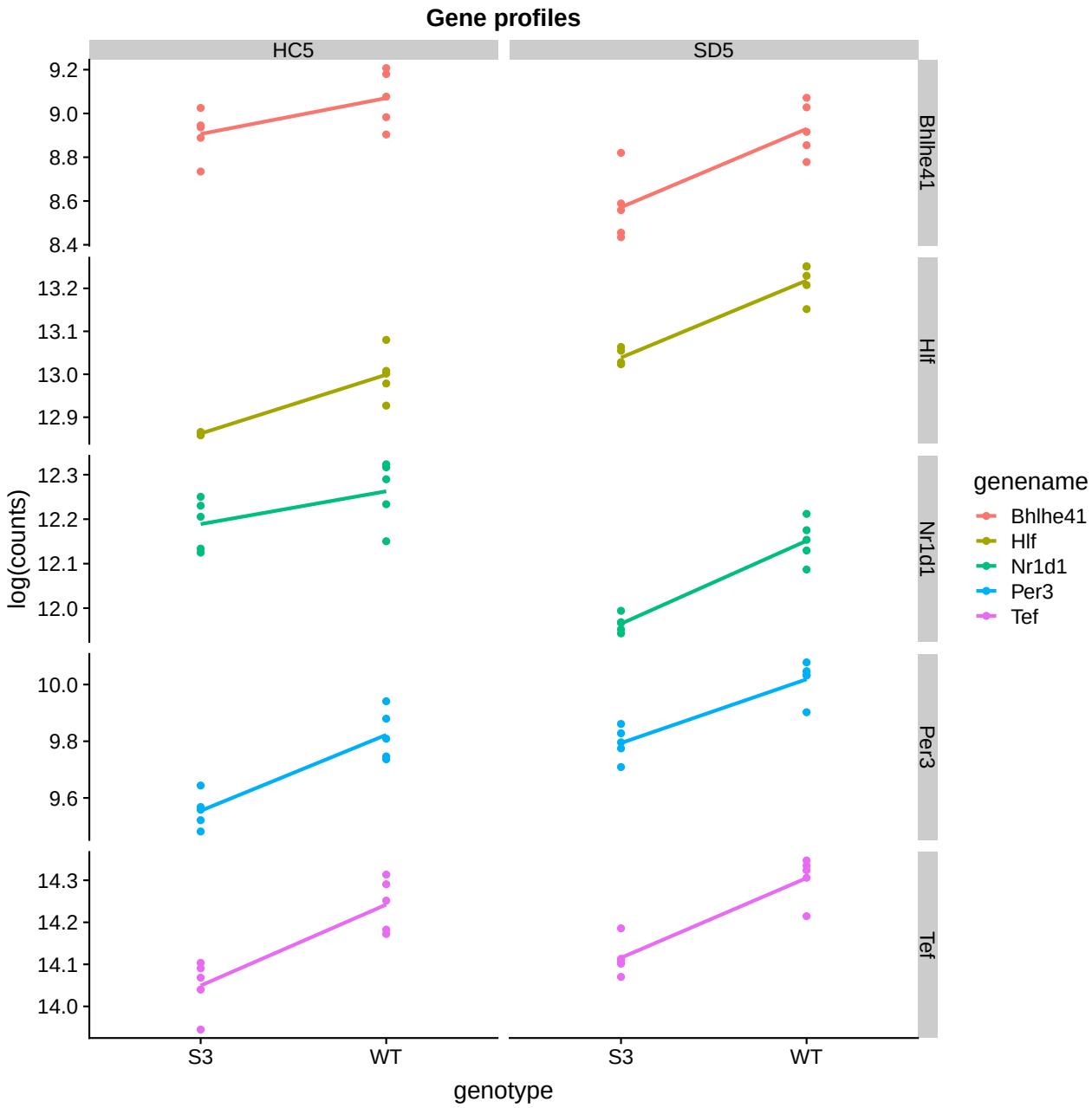
```
            res.o=de.genes.symb, show.plot=TRUE, plotly.flag=FALSE, log.flag=TRUE)
print(g)
```




```
save_plot(filename=paste0("plots/", "Nr1d1_Hlf_Per3_Bhlhe41_Tef", "_log_gene_profile_genotype.pdf"), pl
        base_height=15, base_width=15)
```

Group gene profiles by condition

```
g <- geneGroupProfileRowsRev(normalized.counts=normExprData, design.matrix=designMatrix,
            gene.names=c("Nr1d1", "Hlf", "Per3", "Bhlhe41", "Tef"),
            res.o=de.genes.symb, show.plot=TRUE, plotly.flag=FALSE, log.flag=TRUE)
print(g)
```




```
save_plot(filename=paste0("plots/", "Nr1d1_Hlf_Per3_Bhlhe41_Tef", "_log_gene_profile_condition.pdf"), pl
        base_height=15, base_width=15)
```