Load Libraries

```
library("DESeq2") # package for DESeq run
library("combinat")
library("BiocParallel") # for parallel evaluation
library("dplyr") # for data compilation
library("ggplot2") # plotting
library("vsn") #
library("pheatmap") # for heatmap plotting
library("RColorBrewer") # provides several color palette
```

data preprocessing

```
# read in counts dataset as a matrix
cts <- as.matrix(read.csv("PC9_expression_matrix.csv", row.names = 1))
# read in annotation dataset, and use the first column as rownames
coldata <- read.csv("colData.csv", row.names=1)
# subset 2 columns: condition and type
coldata <- coldata[,c("file_id", "condition", "read_type")]
# convert type of items in condition into 'factor' type.
coldata$condition <- factor(coldata$condition)
# convert type of items in type into 'factor' type.
coldata$read_type <- factor(coldata$read_type)
# remove the character 'fb' in rownames of coldata
rownames(coldata) <- coldata$file_id
# subset the cts by column names included in the row names of 'coldata', and reorder them in order of c
cts <- cts[, rownames(coldata)]
cts <- round(cts)
```

1.Differential expression analysis

```
# generate metadata for multi-comparison
metadata <- data.frame(Untreated = coldata$file_id[1:3],
                       Day3 = coldata$file_id[4:6],
                       Day9 = coldata$file_id[7:9])
# run mult-
multi <- DESeq_multi_group(exprSet = cts, meta = metadata, coldata = coldata)
```

```
## converting counts to integer mode
## factor levels were dropped which had no samples
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
```

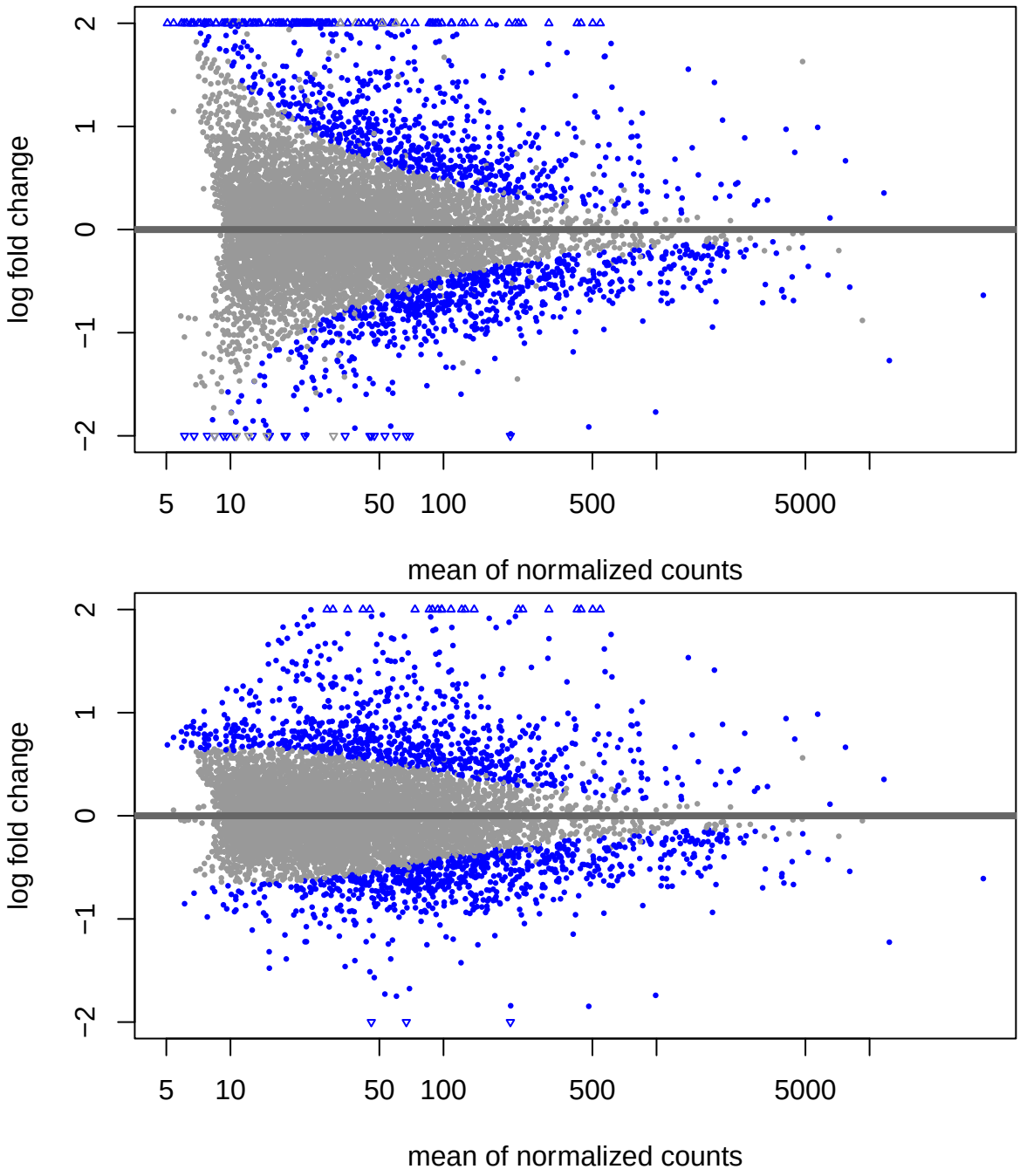
```
## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2 vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895

## converting counts to integer mode

## factor levels were dropped which had no samples

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2 vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895

## converting counts to integer mode

## factor levels were dropped which had no samples

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2 vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895
```

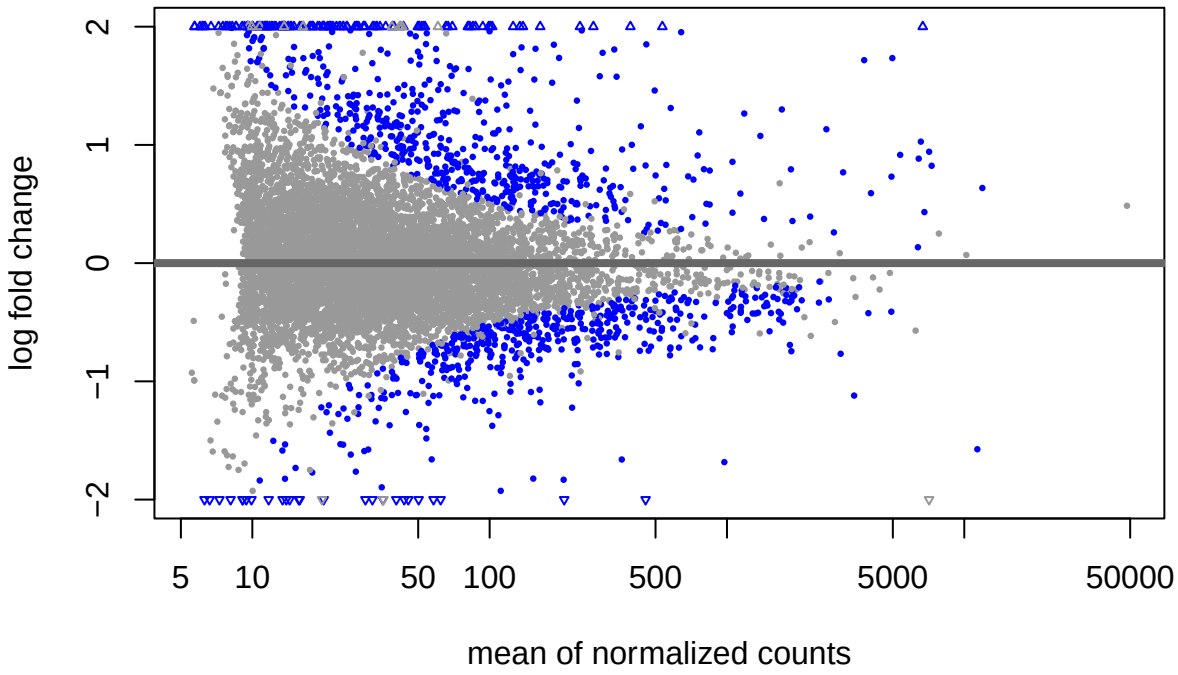
```
dds_list <- multi$dds_list
results_list <- multi$results_list
resLFC_list <- multi$resLFC_list
```

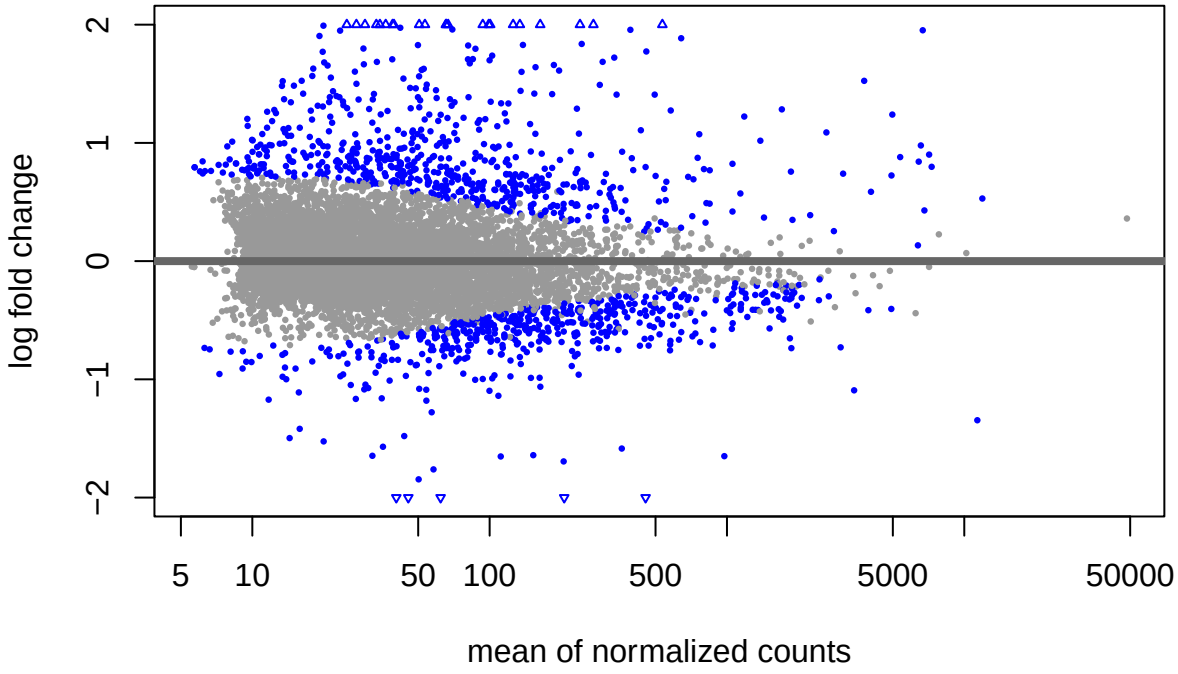
–>

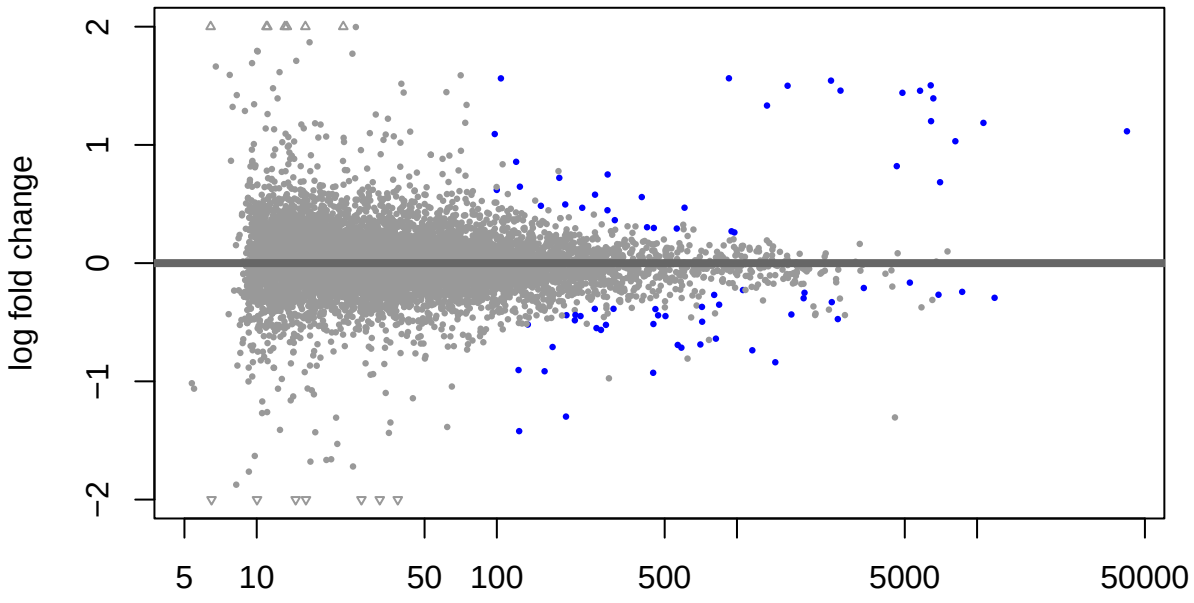
2. Exploring and exporting results

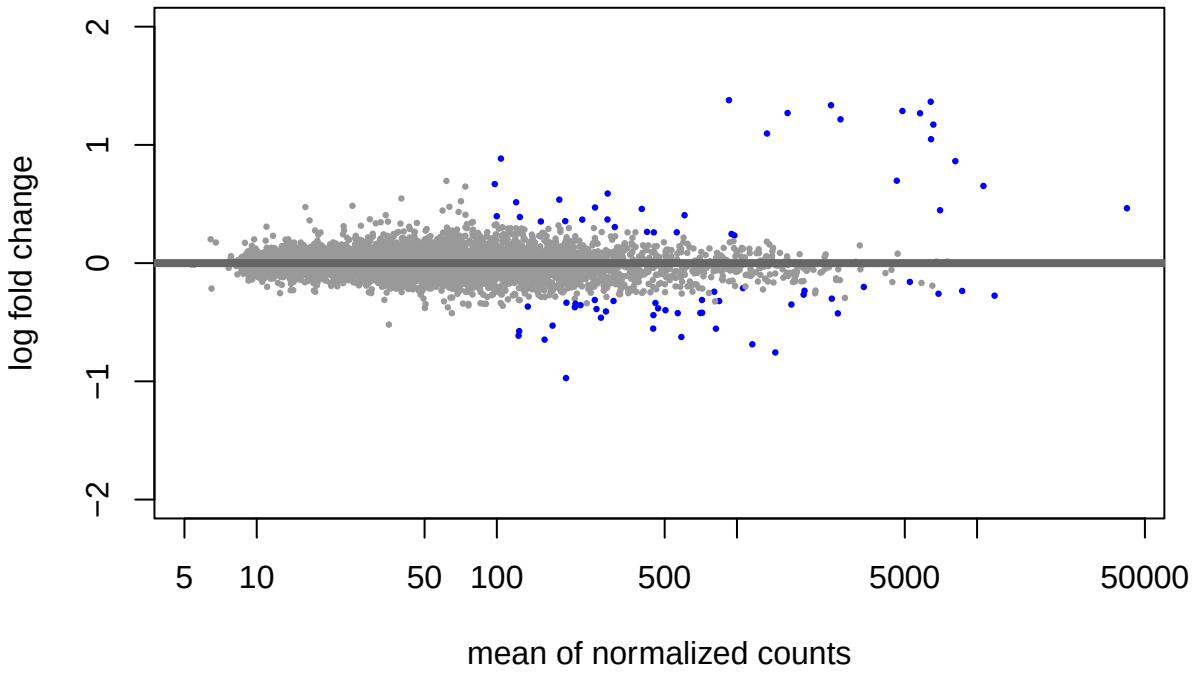
```
for(i in 1:length(results_list)){
  plotMA(results_list[[i]], ylim=c(-2,2)) # MA-plot
  plotMA(resLFC_list[[i]], ylim=c(-2,2)) # MA-plot for log FC data
}
```


log fold change
2
1
0
−1
−2
5
10
50
100
500
5000
mean of normalized counts
log fold change
2
1
0
−1
−2
5
10
50
100
500
5000
mean of normalized counts


log fold change
2
1
0
−1
−2
5
10
50
100
500
5000
50000
mean of normalized counts


log fold change
2
1
0
−1
−2
5
10
50
100
500
5000
50000
mean of normalized counts






```
# interaction with MAplot
# idx <- identify(res$baseMean, res$log2FoldChange) # get to identify MA-plot
# rownames(res)[idx]
```

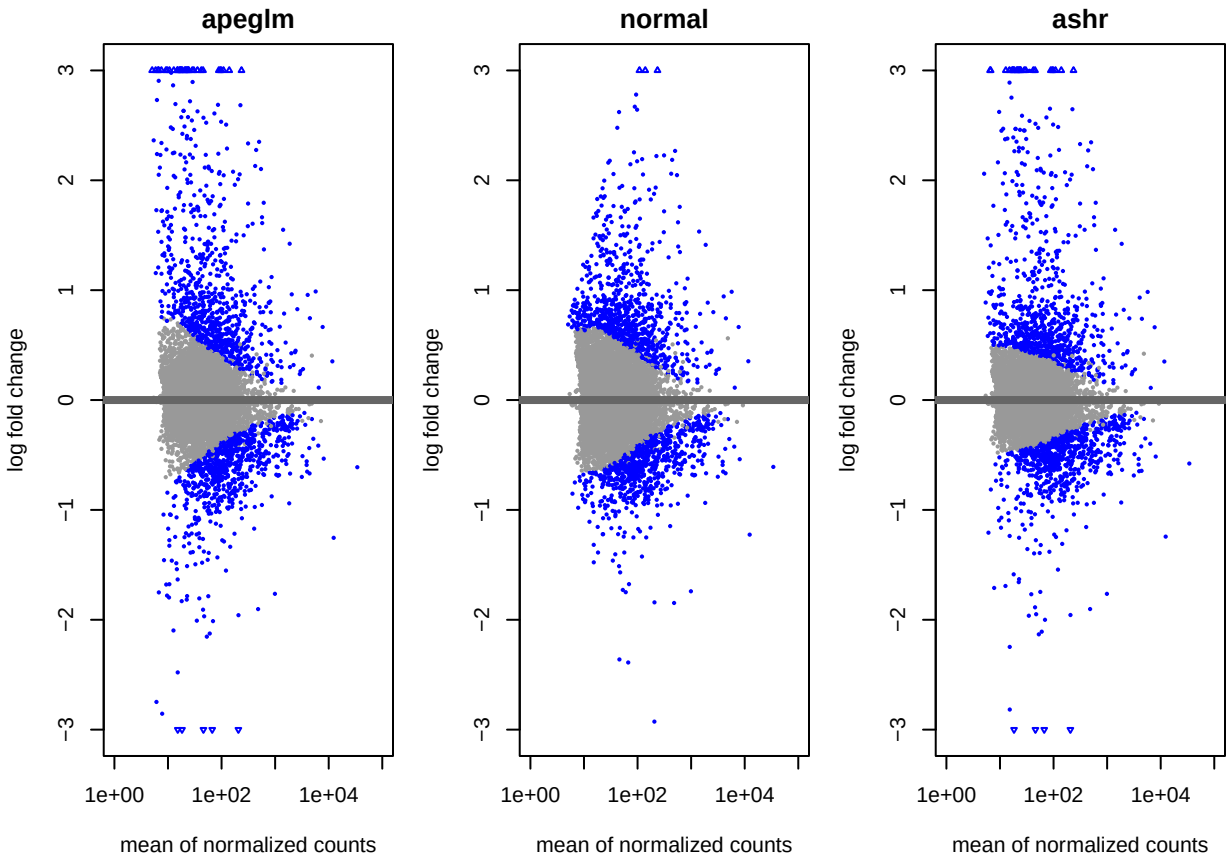
```
# because we are interested in treated vs untreated, we set 'coef=2'
resNorm_list <- list()
resApm_list <- list()
resAsh_list <- list()
for(i in 1:length(results_list)){
  resNorm <- lfcShrink(dds_list[[i]], coef=2, type="normal") # using 'normal' for LFC shrinkage
  resNorm_list[[names(results_list[i])]] <- resNorm
  resApm <- lfcShrink(dds_list[[i]], coef=2, type="apeglm") # using 'normal' for LFC shrinkage
```

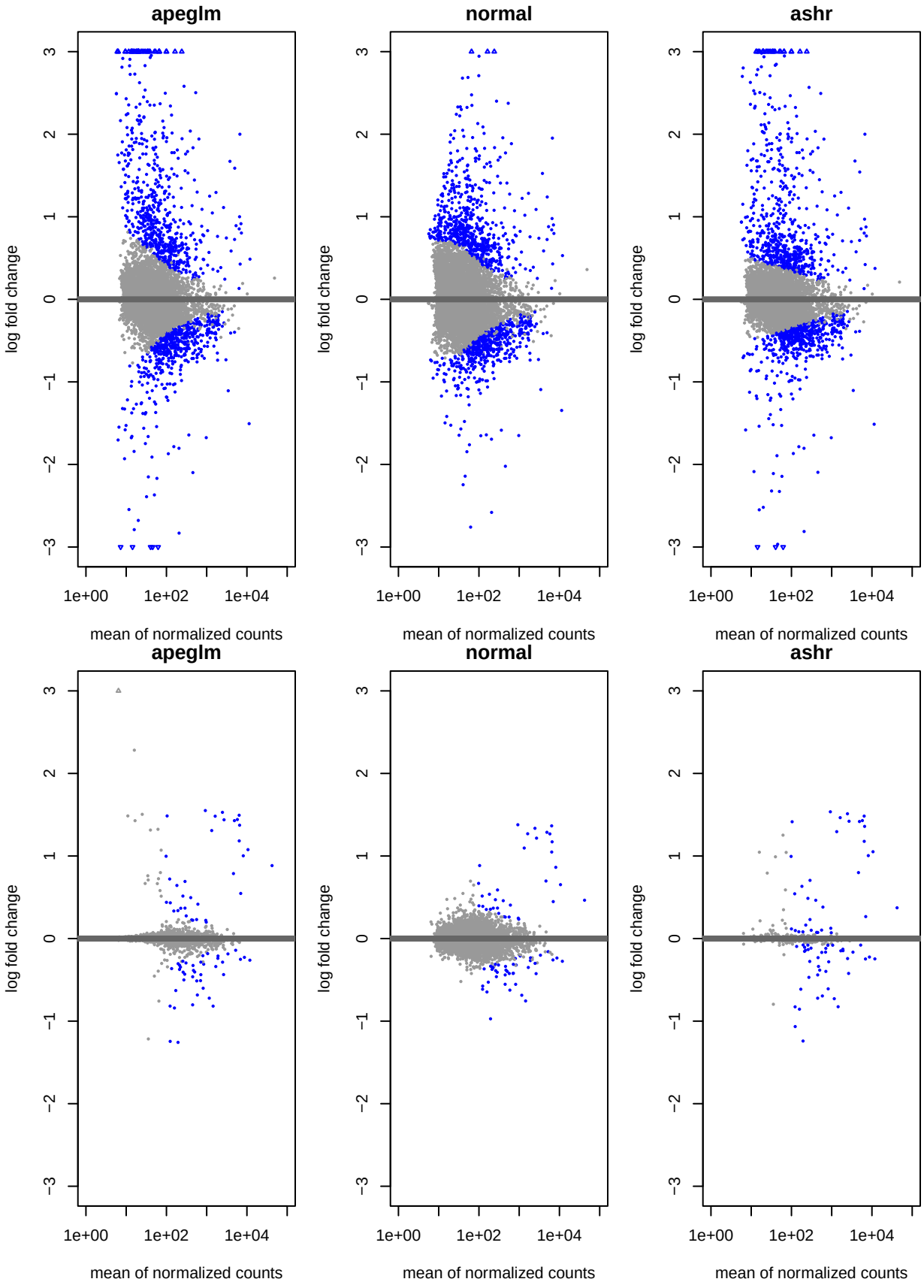
```r
  resApm_list[[names(results_list[i])]] <- resApm
  resAsh <- lfcShrink(dds_list[[i]], coef=2, type="ashr") # using 'ashr' for LFC shrinkage, by which th
  resAsh_list[[names(results_list[i])]] <- resAsh
}
```

```
## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2 vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895

## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895

## using 'ashr' for LFC shrinkage. If used in published research, please cite:
##     Stephens, M. (2016) False discovery rates: a new deal. Biostatistics, 18:2.
##     https://doi.org/10.1093/biostatistics/kxw041

## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2 vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895

## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895

## using 'ashr' for LFC shrinkage. If used in published research, please cite:
##     Stephens, M. (2016) False discovery rates: a new deal. Biostatistics, 18:2.
##     https://doi.org/10.1093/biostatistics/kxw041

## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2 vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895

## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895

## using 'ashr' for LFC shrinkage. If used in published research, please cite:
##     Stephens, M. (2016) False discovery rates: a new deal. Biostatistics, 18:2.
##     https://doi.org/10.1093/biostatistics/kxw041
```

```r
# plot 3 types respectively
for(i in 1:length(results_list)){
  par(mfrow=c(1,3), mar=c(4,4,2,1)) # set margin size and organization
  xlim <- c(1,1e5); ylim <- c(-3,3) # set x/y-axis limits
  # plot 3 MA plots
  plotMA(resApm_list[[i]], xlim=xlim, ylim=ylim, main="apeglm")
  plotMA(resNorm_list[[i]], xlim=xlim, ylim=ylim, main="normal")
```

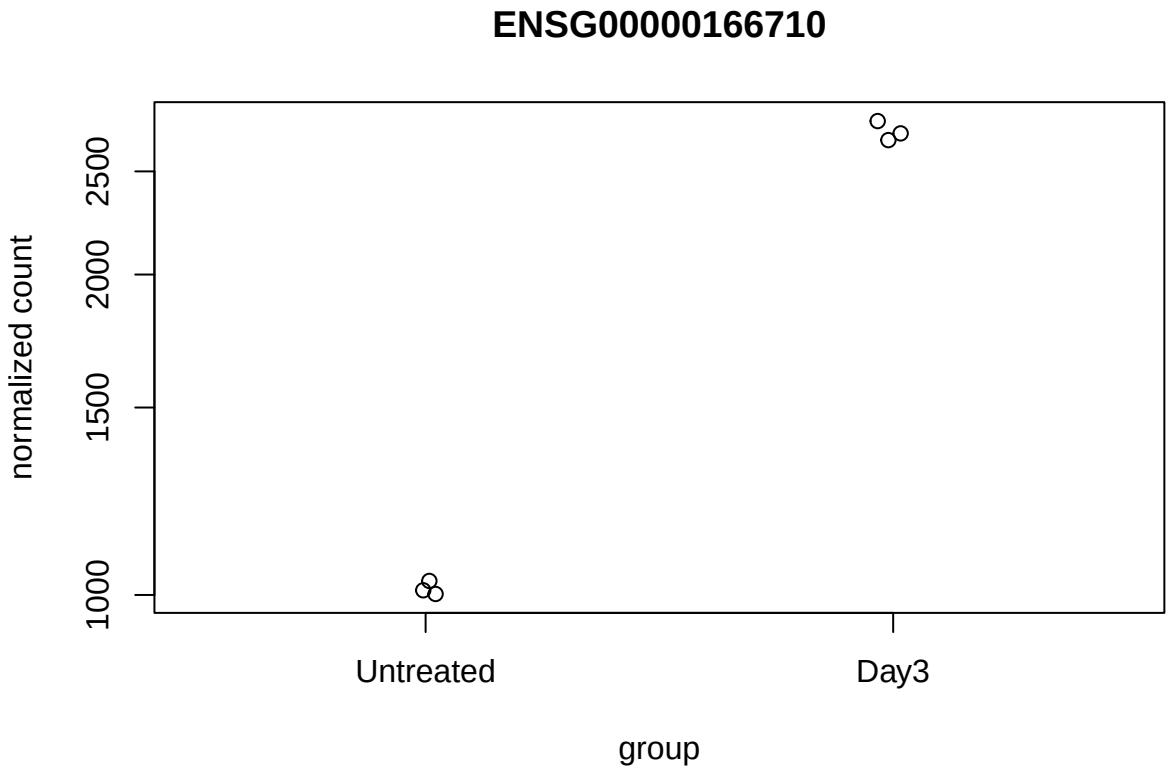
```
  plotMA(resAsh_list[[i]], xlim=xlim, ylim=ylim, main="ashr")
}
```

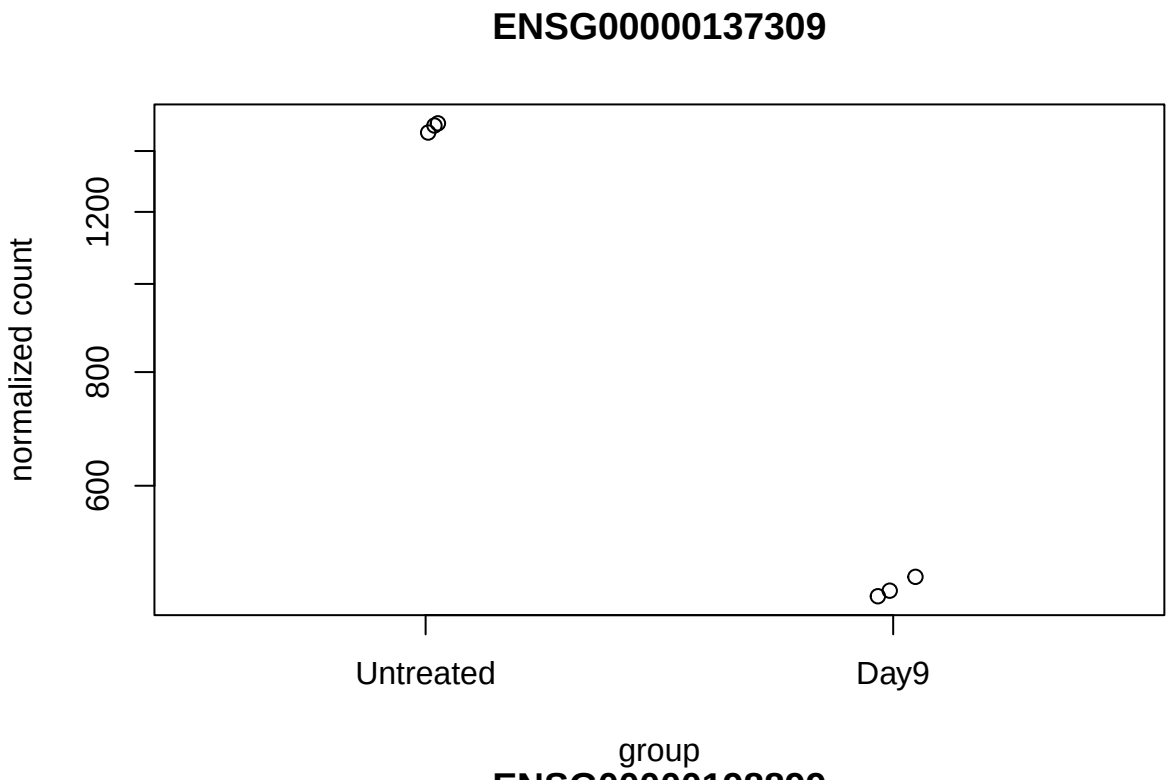


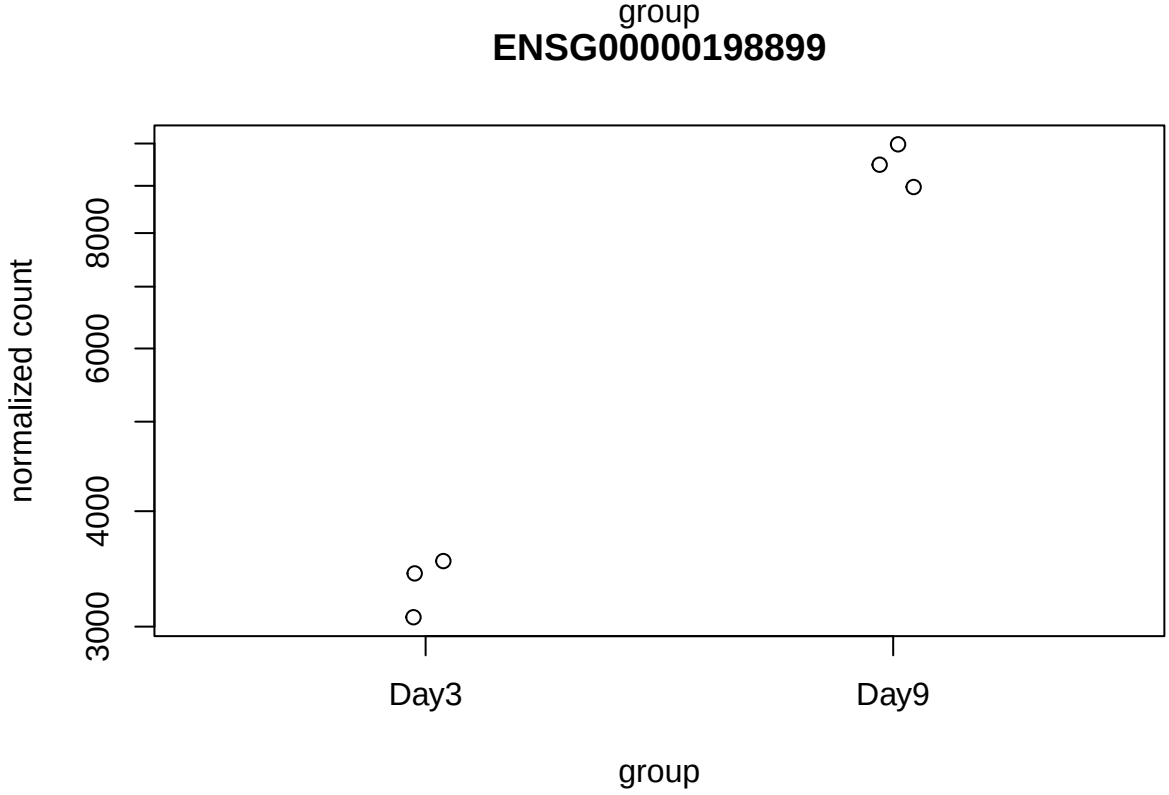

apeglm
normal
ashr
log fold change
mean of normalized counts
1e+00
1e+02
1e+04
3
2
1
0
−1
−2
−3

plot count

```
for(i in 1:length(results_list)){
  plotCounts(dds_list[[i]], gene=which.min(results_list[[i]]$padj), intgroup="condition") # plot count

  d <- plotCounts(dds_list[[i]], gene=which.min(results_list[[i]]$padj), intgroup="condition",
                  returnData=TRUE)
  # use ggolot to demonstrate it
  ggplot(d, aes(x=condition, y=count)) +
    geom_point(position=position_jitter(w=0.1,h=0)) +
    scale_y_log10(breaks=c(25,100,400))
}
```








```
# More information on results columns
mcols(results_list[[i]])$description
```

```
## [1] "mean of normalized counts for all samples"
## [2] "log2 fold change (MLE): condition Day9 vs Day3"
## [3] "standard error: condition Day9 vs Day3"
```

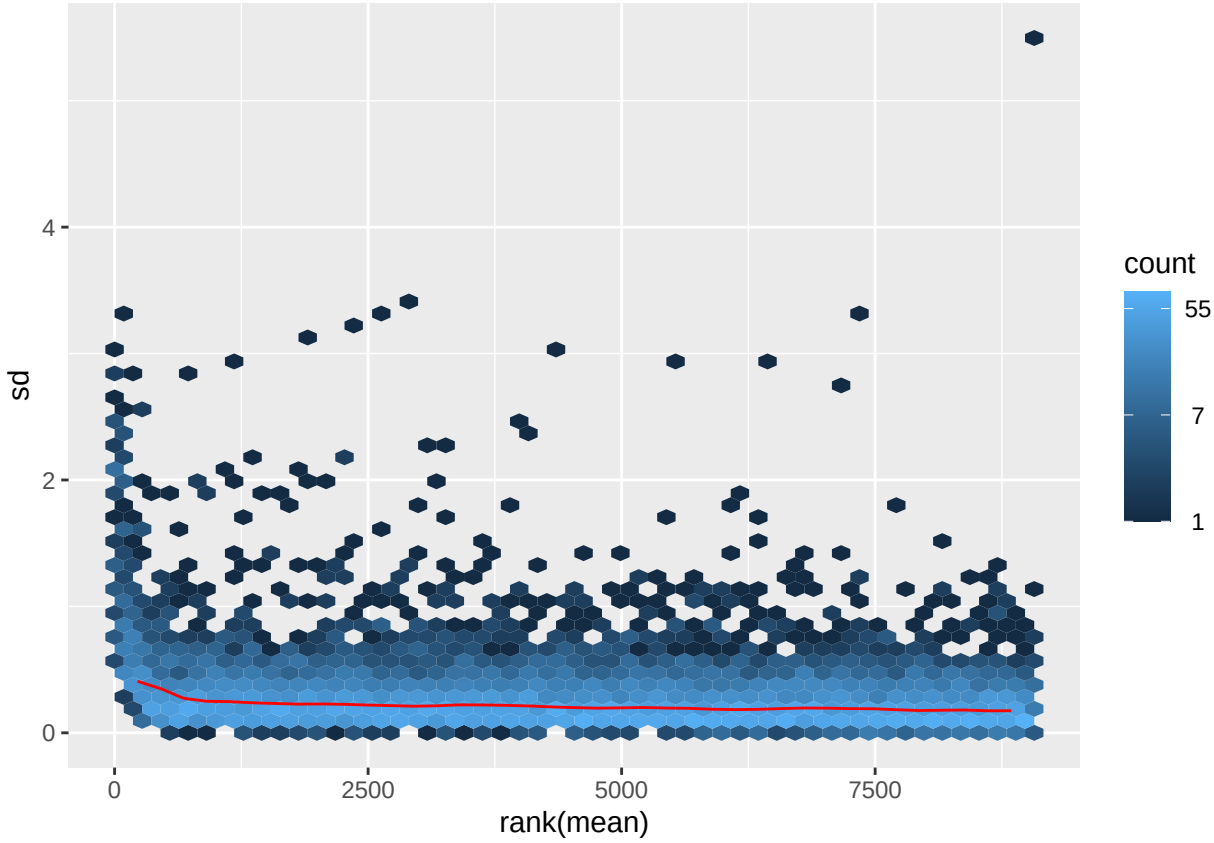
```
## [4] "Wald statistic: condition Day9 vs Day3"
## [5] "Wald test p-value: condition Day9 vs Day3"
## [6] "BH adjusted p-values"
```

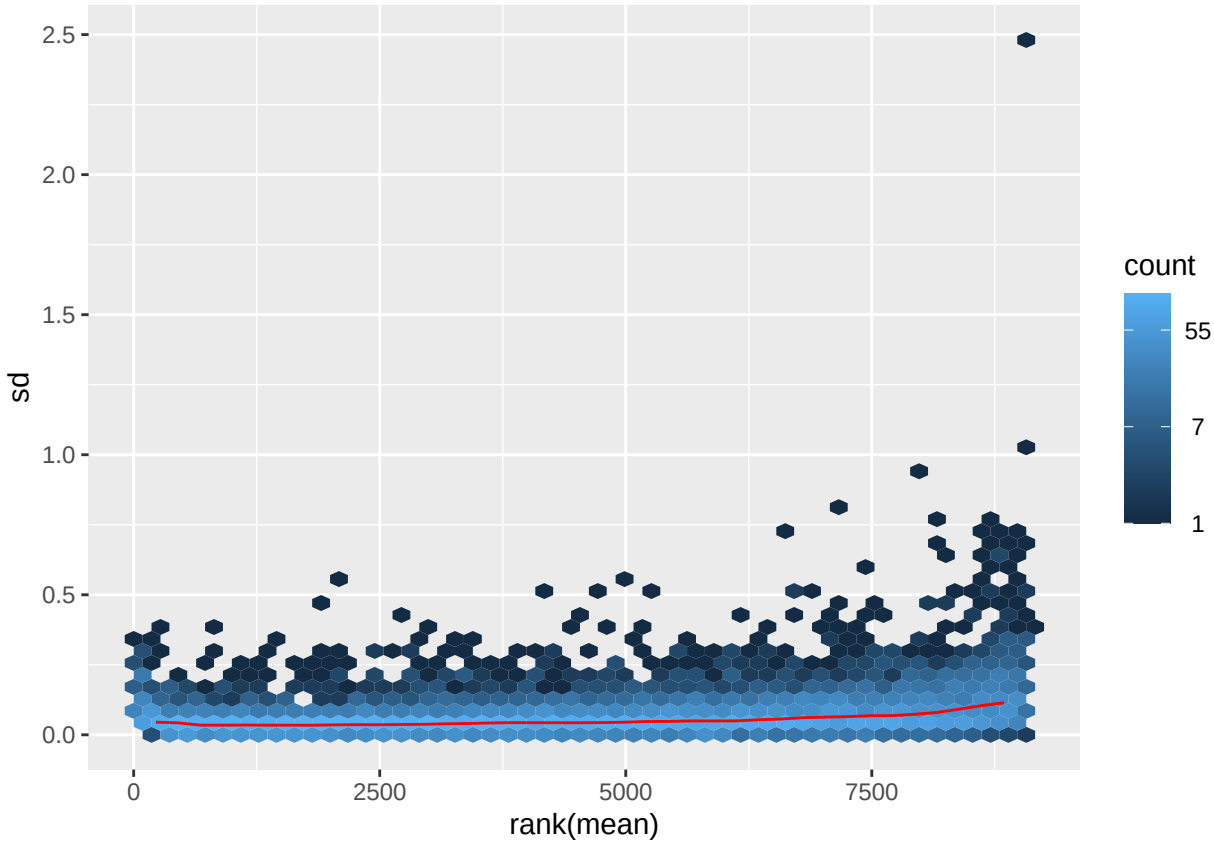
3.Data transformations and visualization

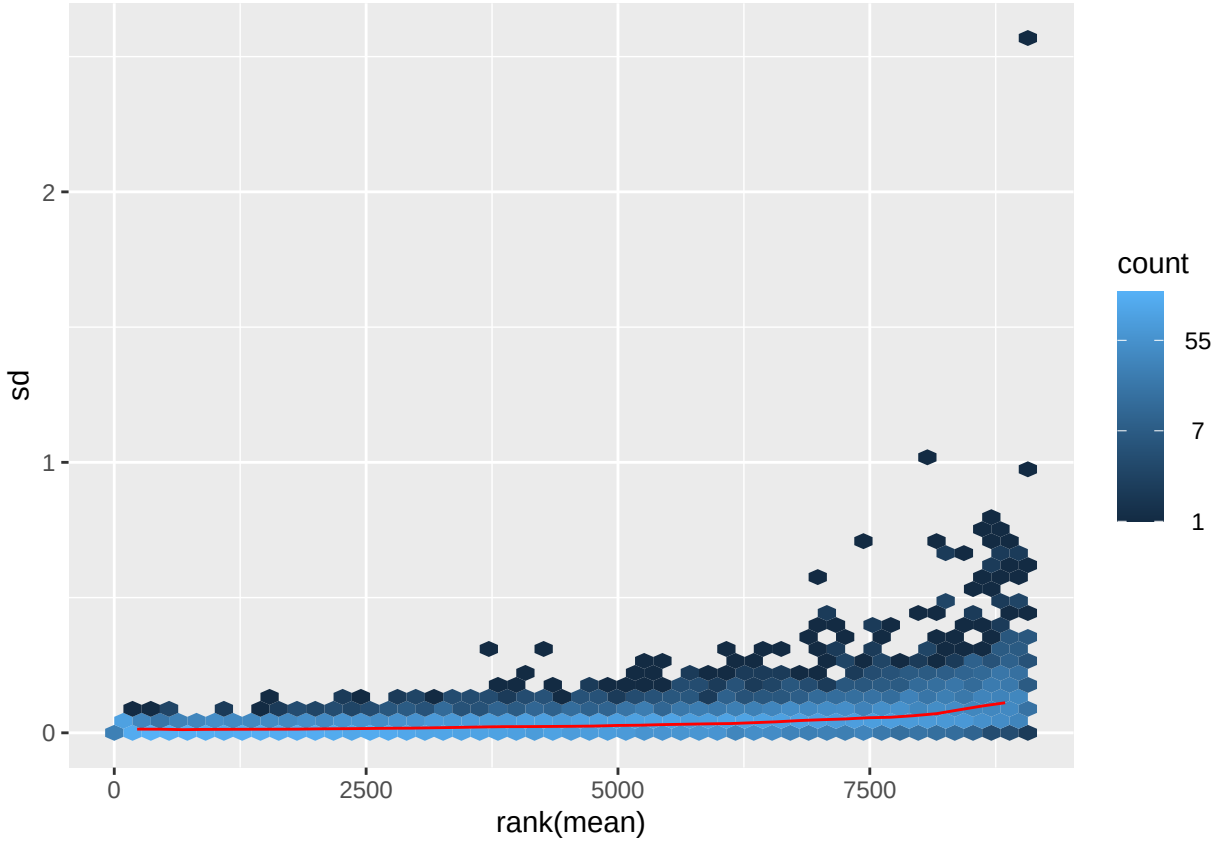
```
vsd_list <- list()
rld_list <- list()
ntd_list <- list()

# Count data transformations
for(i in 1:length(results_list)){
  vsd <- vst(dds_list[[i]], blind=FALSE)  # performing variance stabilizing transformations
  rld <- rlog(dds_list[[i]], blind=FALSE) # Apply regularized log transformation
  ntd <- normTransform(dds_list[[i]])
  head(assay(vsd), 3) # show top 3 rows

  vsd_list[[names(results_list[i])]] <- vsd
  rld_list[[names(results_list[i])]] <- rld
  ntd_list[[names(results_list[i])]] <- ntd
}
```

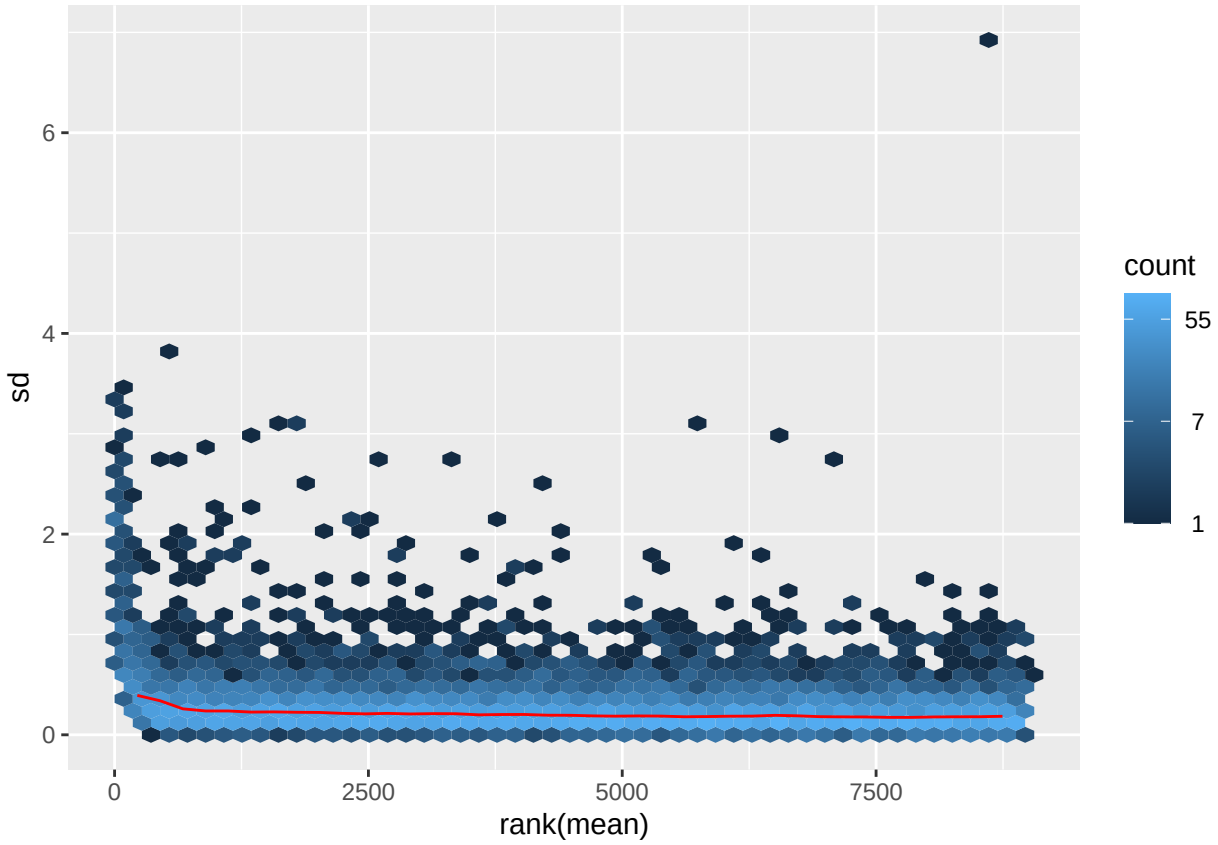
```
## -- note: fitType='parametric', but the dispersion trend was not well captured by the
##    function: y = a/x + b, and a local regression fit was automatically substituted.
##    specify fitType='local' or 'mean' to avoid this message next time.
```

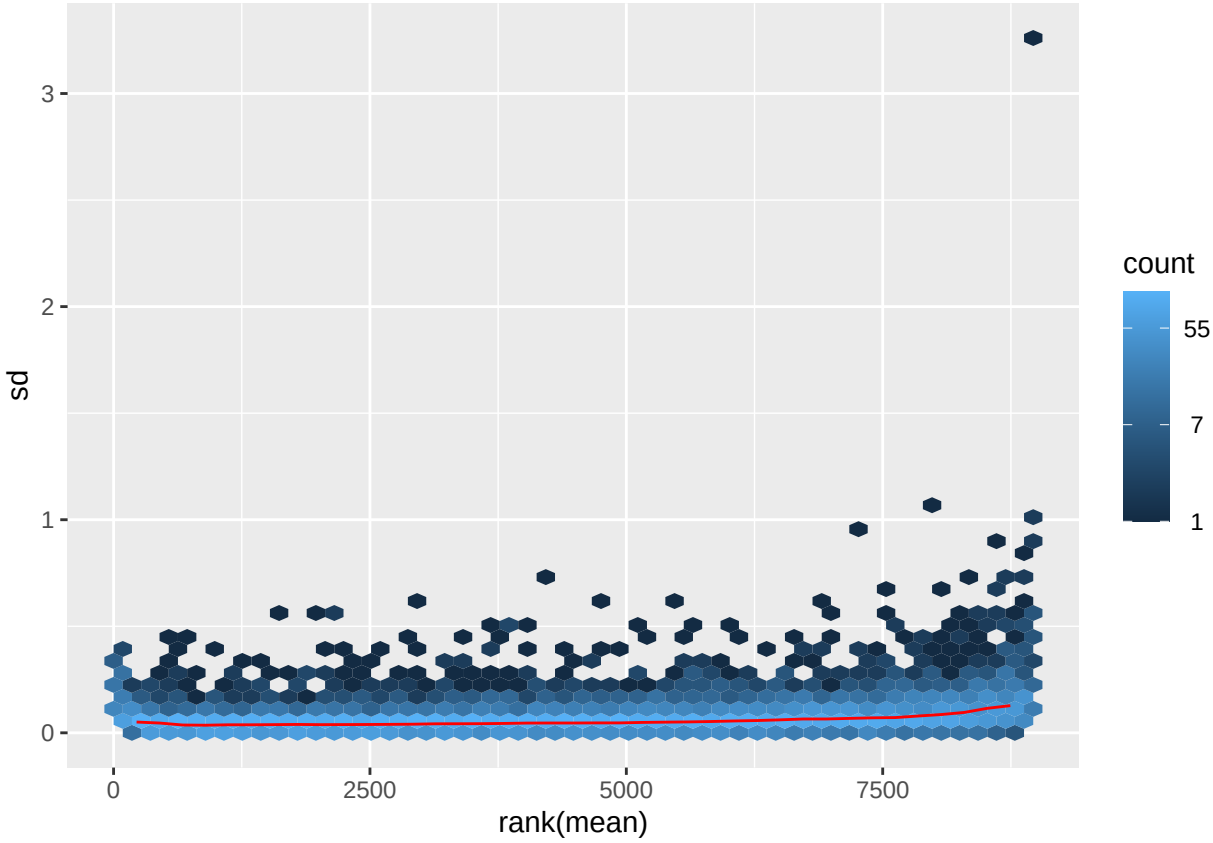
```
library("vsn")
# this gives log2(n + 1)
for(i in 1:length(results_list)){
  meanSdPlot(assay(ntd_list[[i]])) # plot normal transformation
  meanSdPlot(assay(vsd_list[[i]])) # plot variance stabilizing transformations
  meanSdPlot(assay(rld_list[[i]])) # plot r-log
}
```

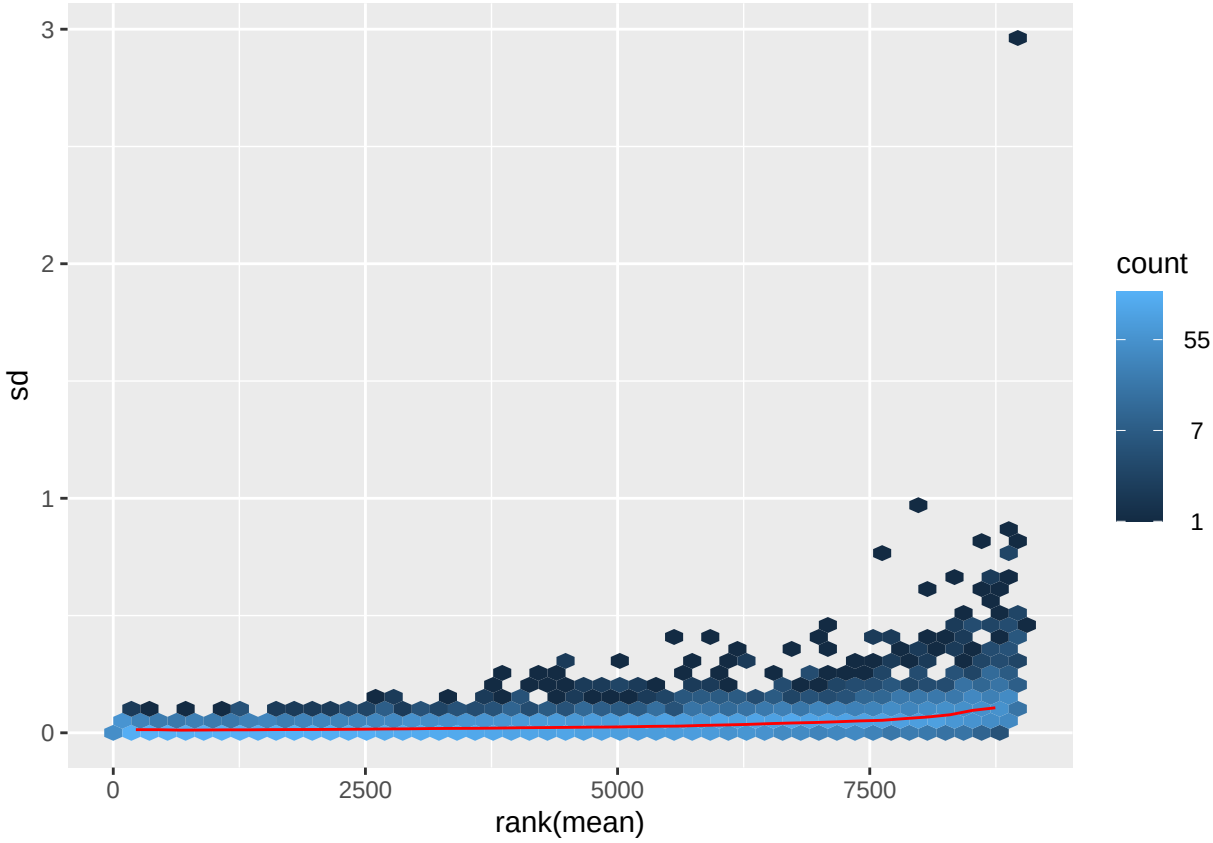

sd
4
2
0
0
2500
5000
7500
rank(mean)
count
55
7
1

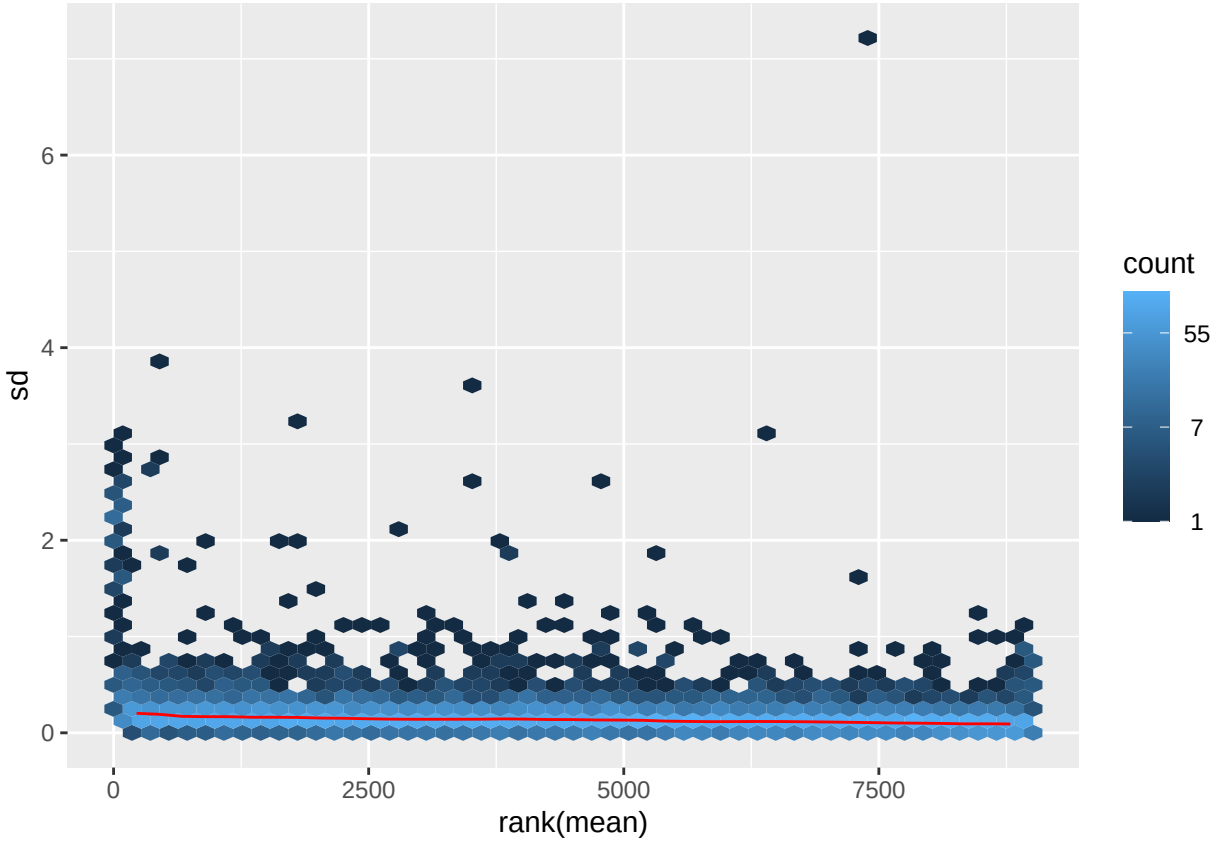

sd
2.5
2.0
1.5
1.0
0.5
0.0
0
2500
5000
7500
rank(mean)
count
55
7
1

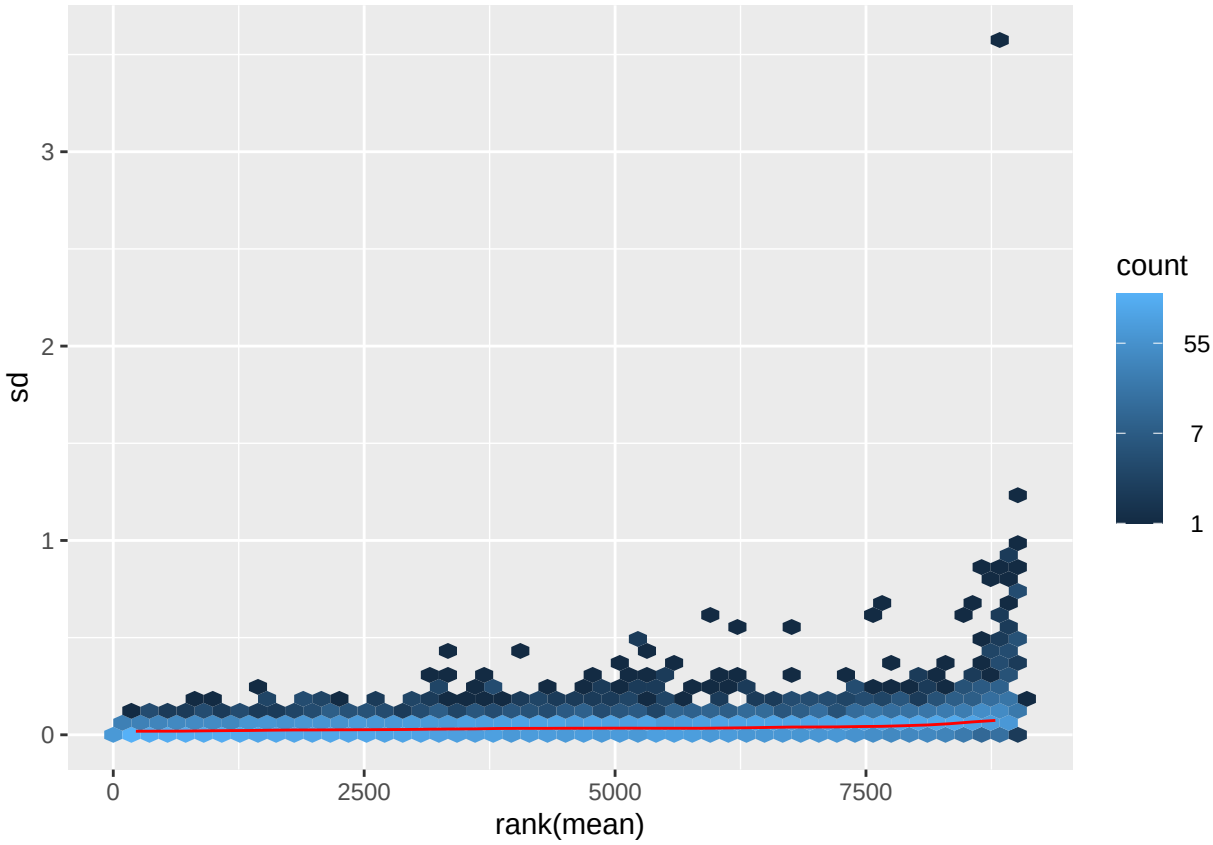

sd
2
1
0
0
2500
5000
7500
rank(mean)
count
55
7
1

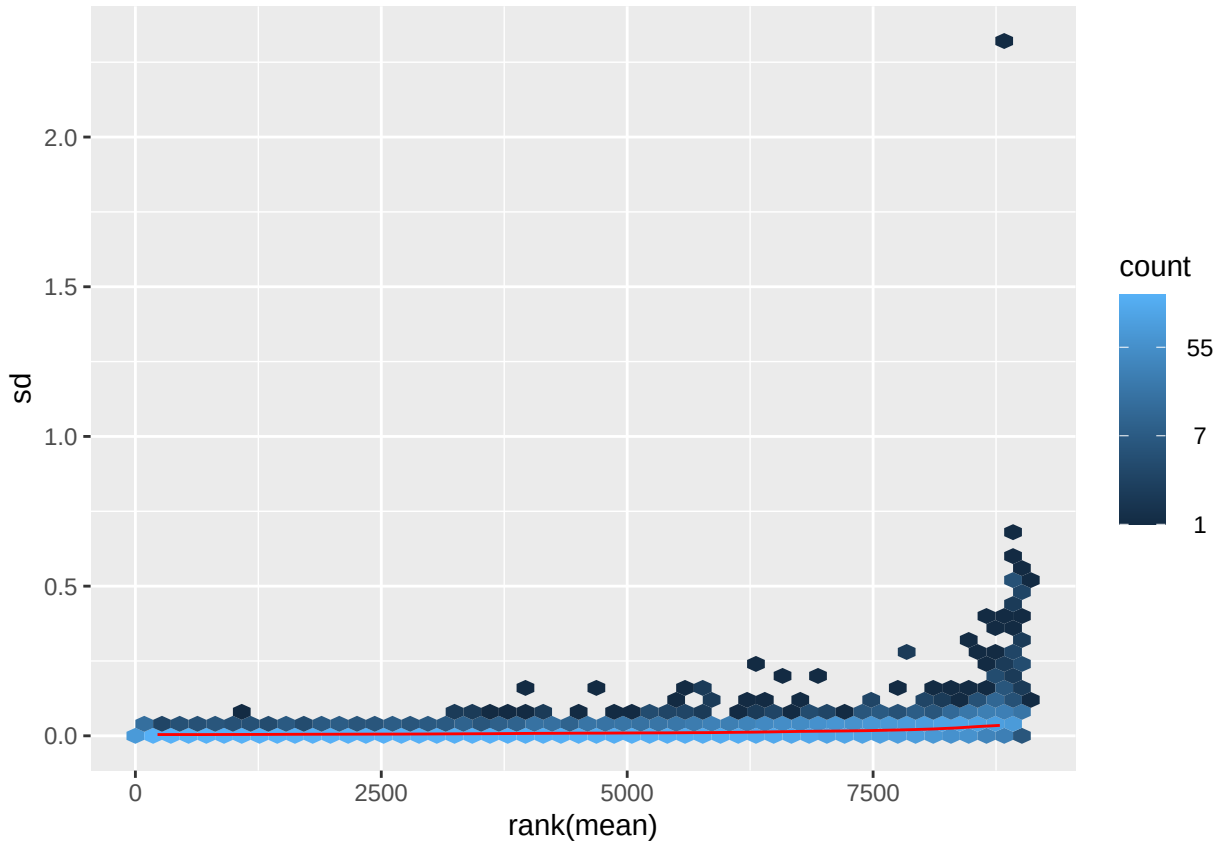

sd
6
4
2
0
0
2500
5000
7500
rank(mean)
count
55
7
1


sd
3
2
1
0
0
2500
5000
7500
rank(mean)
count
55
7
1

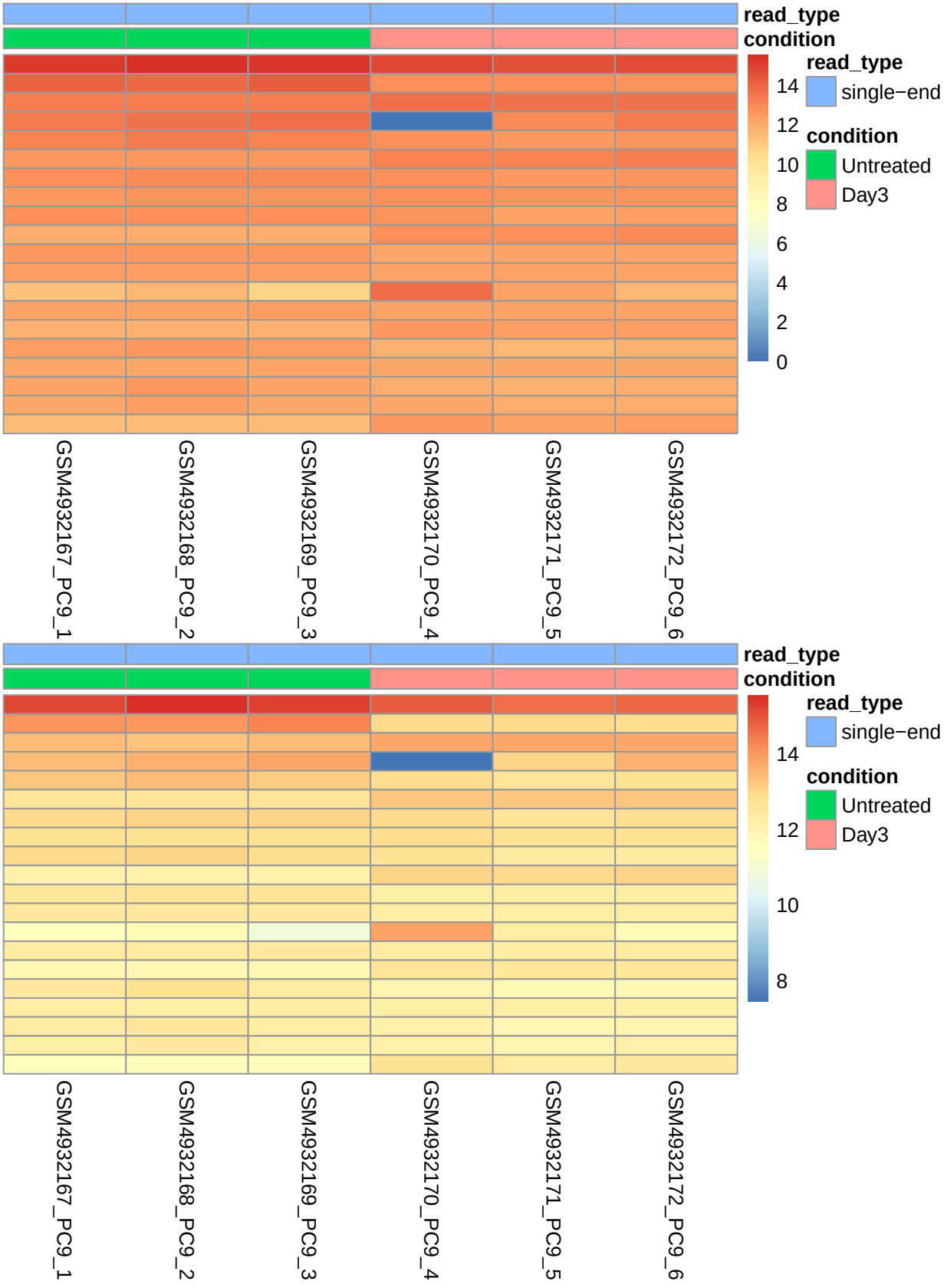

sd
3
2
1
0
0
2500
5000
7500
rank(mean)
count
55
7
1

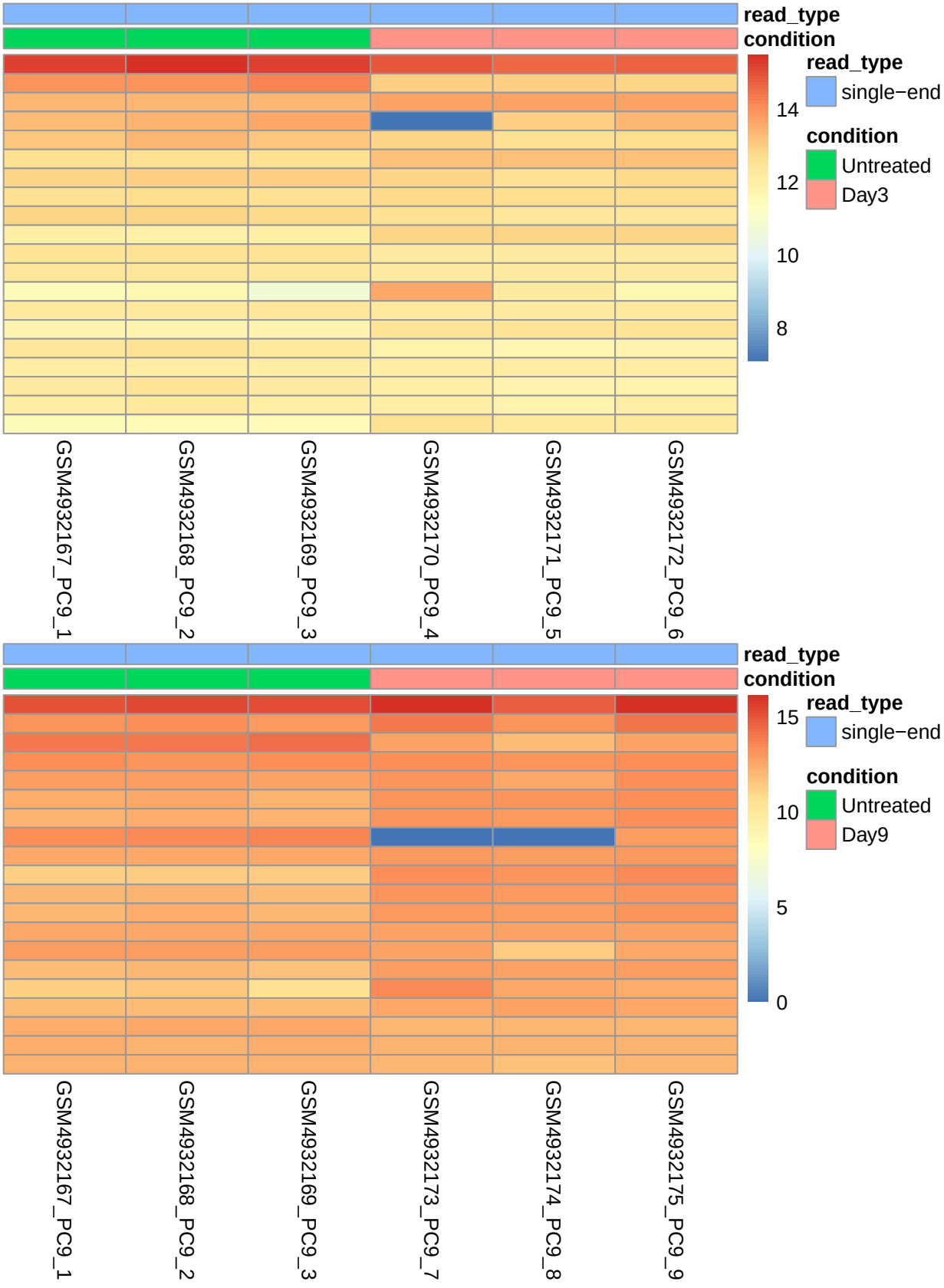

sd
6
4
2
0
0
2500
5000
7500
rank(mean)
count
55
7
1

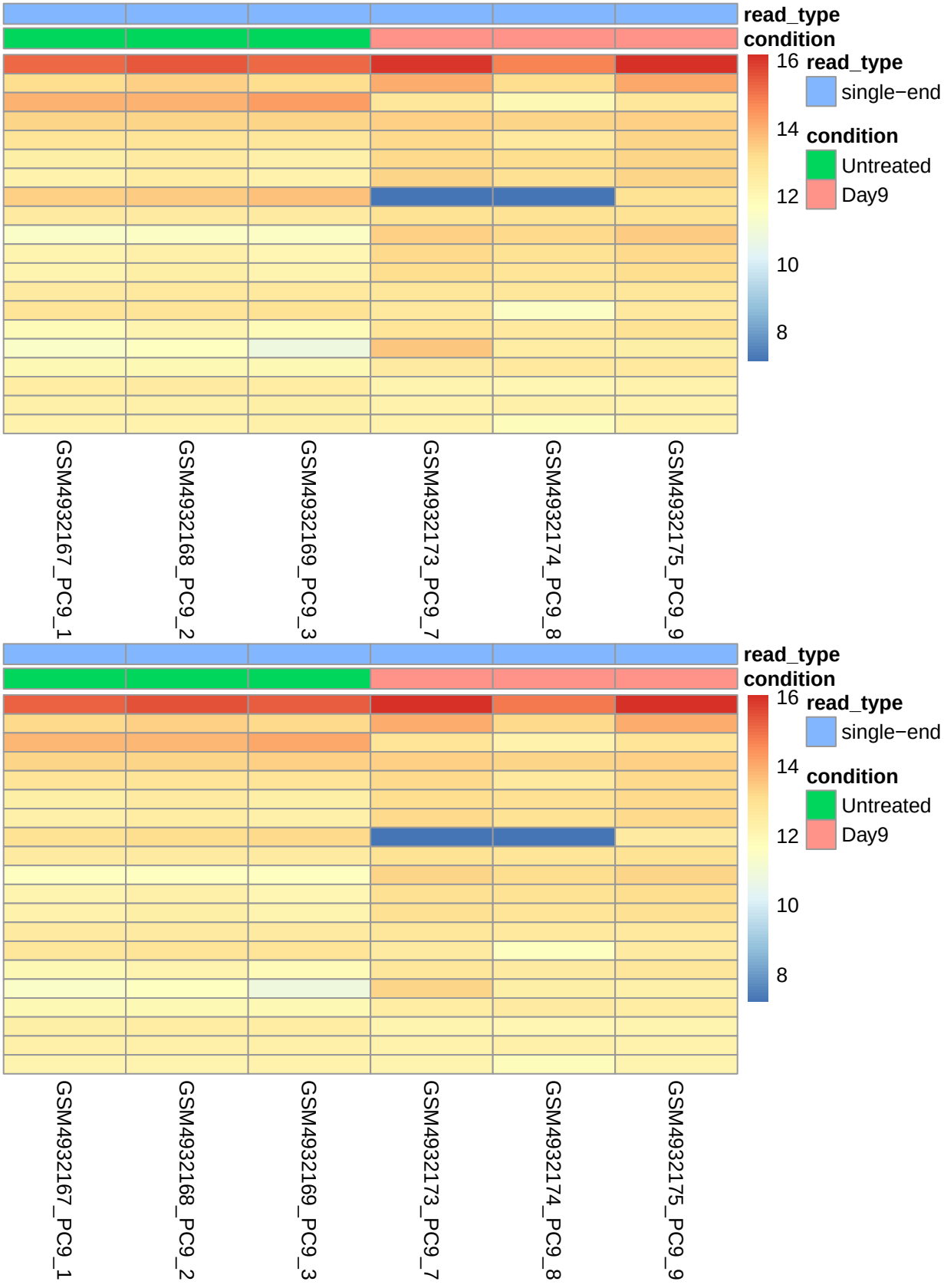

sd
3
2
1
0
0
2500
5000
7500
rank(mean)
count
55
7
1

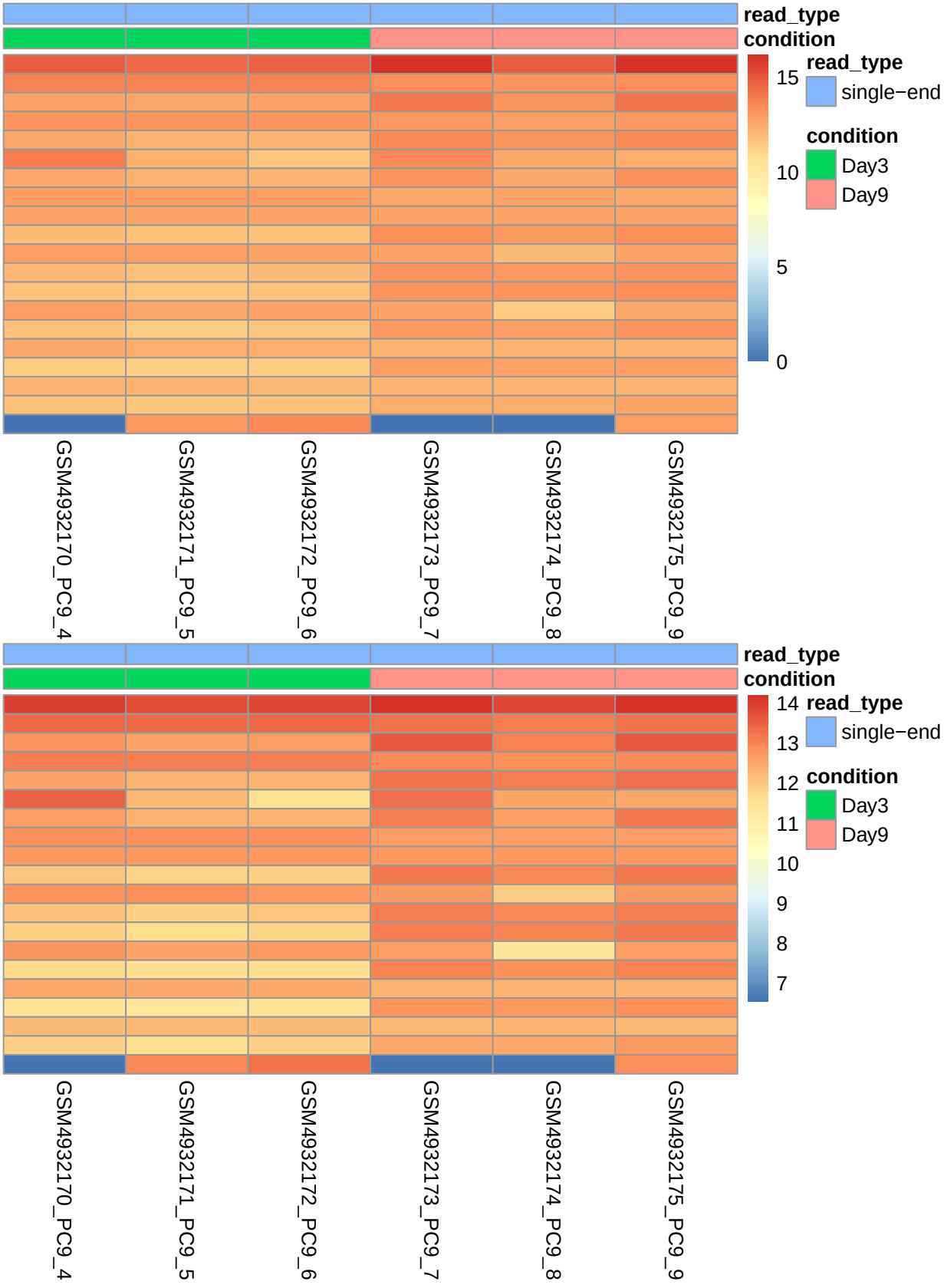


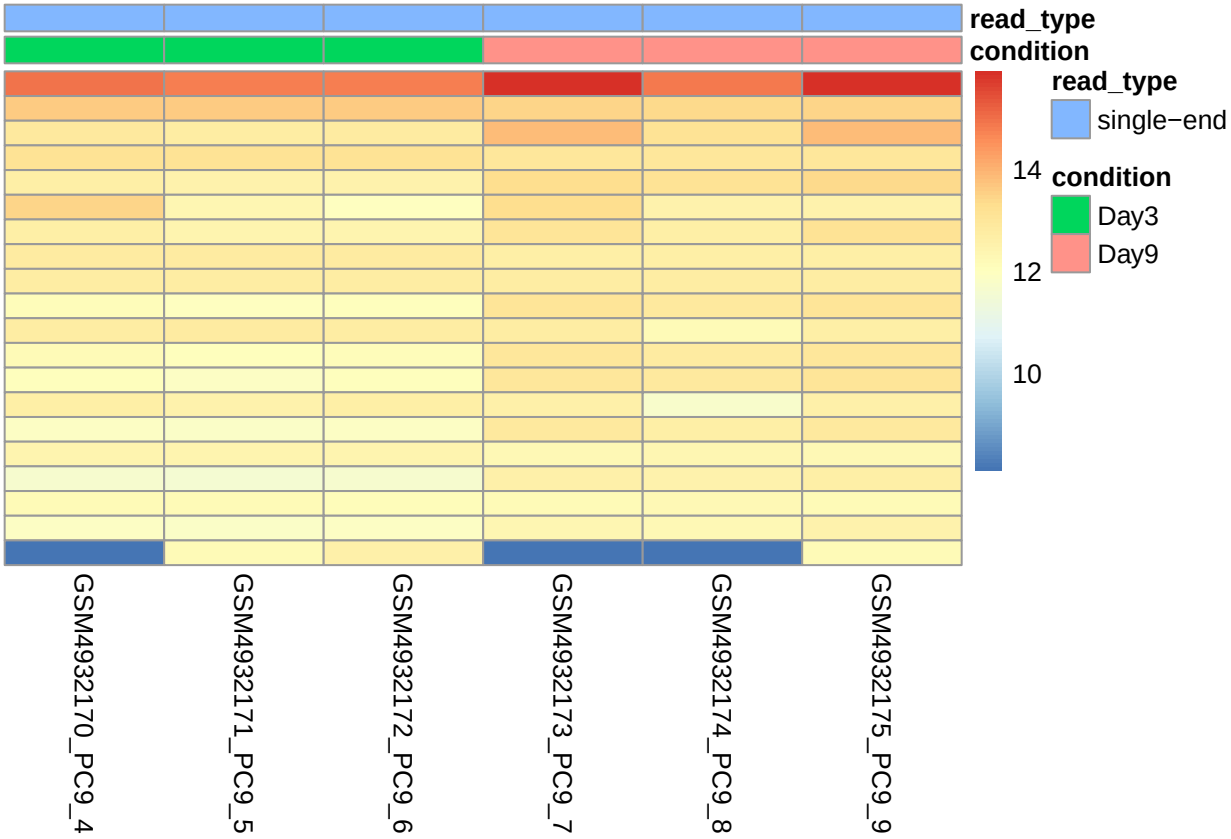

```
  ## Data quality assessment by sample clustering and visualization
  # Heatmap of the count matrix
for(i in 1:length(results_list)){
  select <- order(rowMeans(counts(dds_list[[i]],normalized=TRUE)),
                  decreasing=TRUE)[1:20]
  df <- as.data.frame(colData(dds_list[[i]])[,c("condition","read_type")])
  # heatmap for normal transformation
  pheatmap(assay(ntd_list[[i]])[select,], cluster_rows=FALSE, show_rownames=FALSE,
           cluster_cols=FALSE, annotation_col=df)
  # heatmap for variance stabilizing transformations
  pheatmap(assay(vsd_list[[i]])[select,], cluster_rows=FALSE, show_rownames=FALSE,
           cluster_cols=FALSE, annotation_col=df)
  # heatmap for r-log
  pheatmap(assay(rld_list[[i]])[select,], cluster_rows=FALSE, show_rownames=FALSE,
           cluster_cols=FALSE, annotation_col=df)
}
```


read_type
condition
14
12
10
8
6
4
2
0
read_type
single−end
condition
Untreated
Day3
GSM4932167_PC9_1
GSM4932168_PC9_2
GSM4932169_PC9_3
GSM4932170_PC9_4
GSM4932171_PC9_5
GSM4932172_PC9_6
read_type
condition
14
12
10
8
read_type
single−end
condition
Untreated
Day3
GSM4932167_PC9_1
GSM4932168_PC9_2
GSM4932169_PC9_3
GSM4932170_PC9_4
GSM4932171_PC9_5
GSM4932172_PC9_6

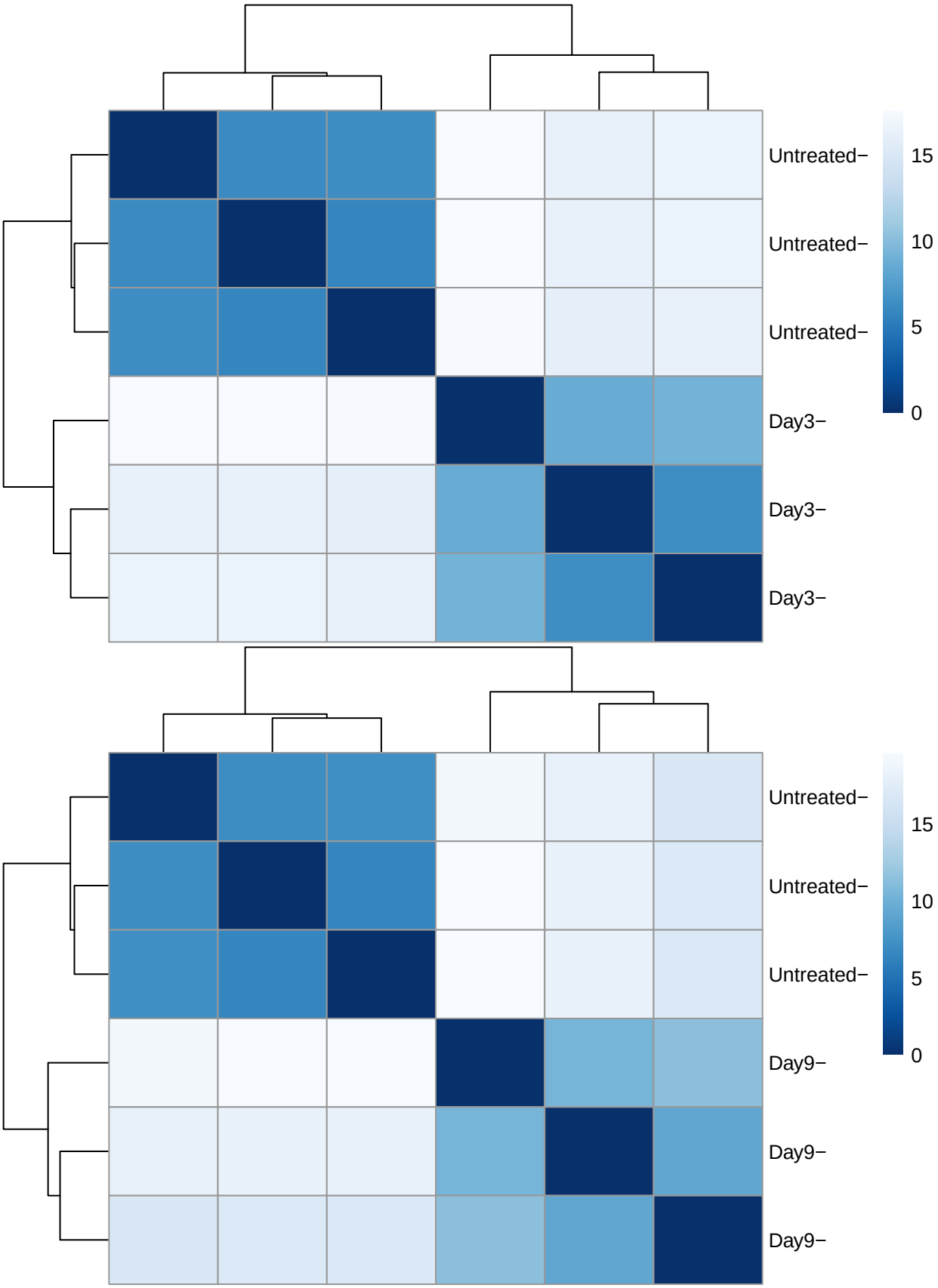

read_type
condition
read_type
single−end
condition
Untreated
Day3
14
12
10
8
GSM4932167_PC9_1
GSM4932168_PC9_2
GSM4932169_PC9_3
GSM4932170_PC9_4
GSM4932171_PC9_5
GSM4932172_PC9_6
read_type
condition
read_type
single−end
condition
Untreated
Day9
15
10
5
0
GSM4932167_PC9_1
GSM4932168_PC9_2
GSM4932169_PC9_3
GSM4932173_PC9_7
GSM4932174_PC9_8
GSM4932175_PC9_9

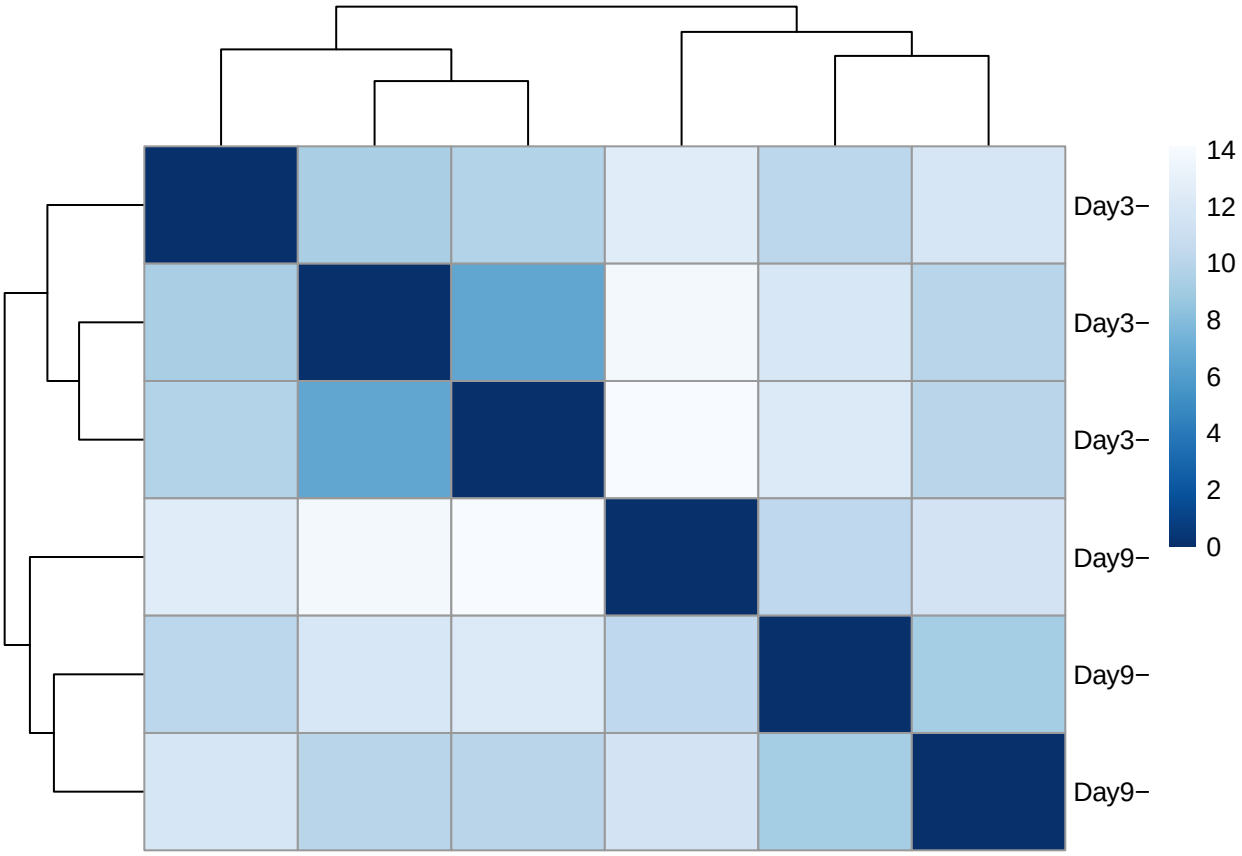

read_type
condition
16
14
12
10
8
read_type
single−end
condition
Untreated
Day9
GSM4932167_PC9_1
GSM4932168_PC9_2
GSM4932169_PC9_3
GSM4932173_PC9_7
GSM4932174_PC9_8
GSM4932175_PC9_9


read_type
condition
15
10
5
0
read_type
single−end
condition
Day3
Day9
GSM4932170_PC9_4
GSM4932171_PC9_5
GSM4932172_PC9_6
GSM4932173_PC9_7
GSM4932174_PC9_8
GSM4932175_PC9_9
read_type
condition
14
13
12
11
10
9
8
7
read_type
single−end
condition
Day3
Day9
GSM4932170_PC9_4
GSM4932171_PC9_5
GSM4932172_PC9_6
GSM4932173_PC9_7
GSM4932174_PC9_8
GSM4932175_PC9_9



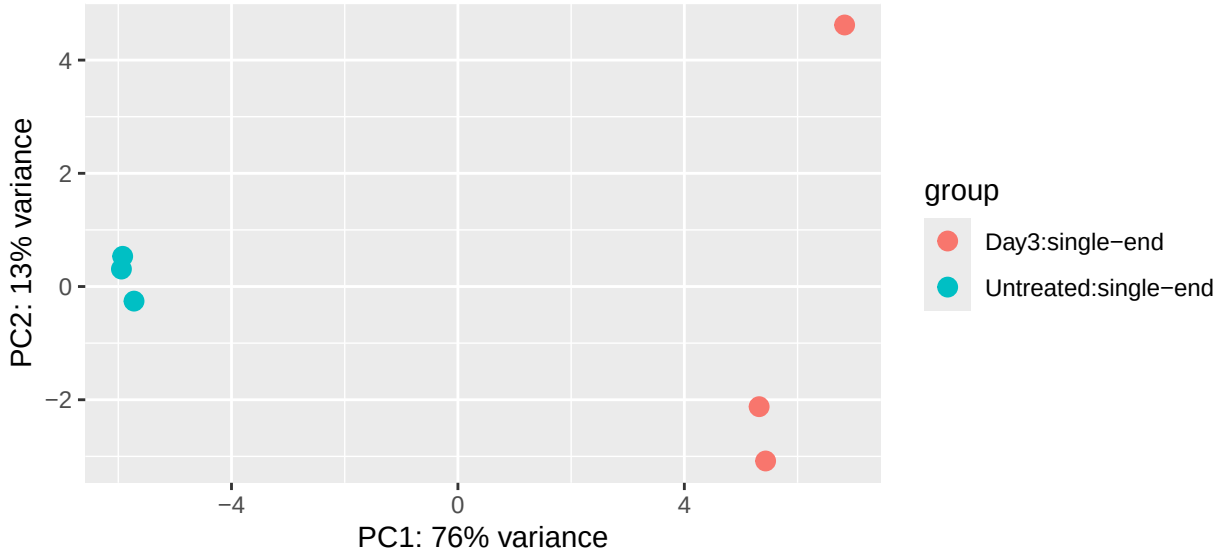

```
for(i in 1:length(results_list)){
  ## plot heatmap of the sample-to-sample distances
  sampleDists <- dist(t(assay(vsd_list[[i]]))) # get distances
  # make it matrix
  sampleDistMatrix <- as.matrix(sampleDists)
  # data preparation
  rownames(sampleDistMatrix) <- paste(vsd_list[[i]]$condition, vsd_list[[i]]$type, sep="-")
  colnames(sampleDistMatrix) <- NULL
  colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255) # set colors
  # plot heatmap for distances
  pheatmap(sampleDistMatrix,
           clustering_distance_rows=sampleDists,
           clustering_distance_cols=sampleDists,
           col=colors)
}
```


Untreated−
Untreated−
Untreated−
Day3−
Day3−
Day3−
15
10
5
0
Untreated−
Untreated−
Untreated−
Day9−
Day9−
Day9−
15
10
5
0



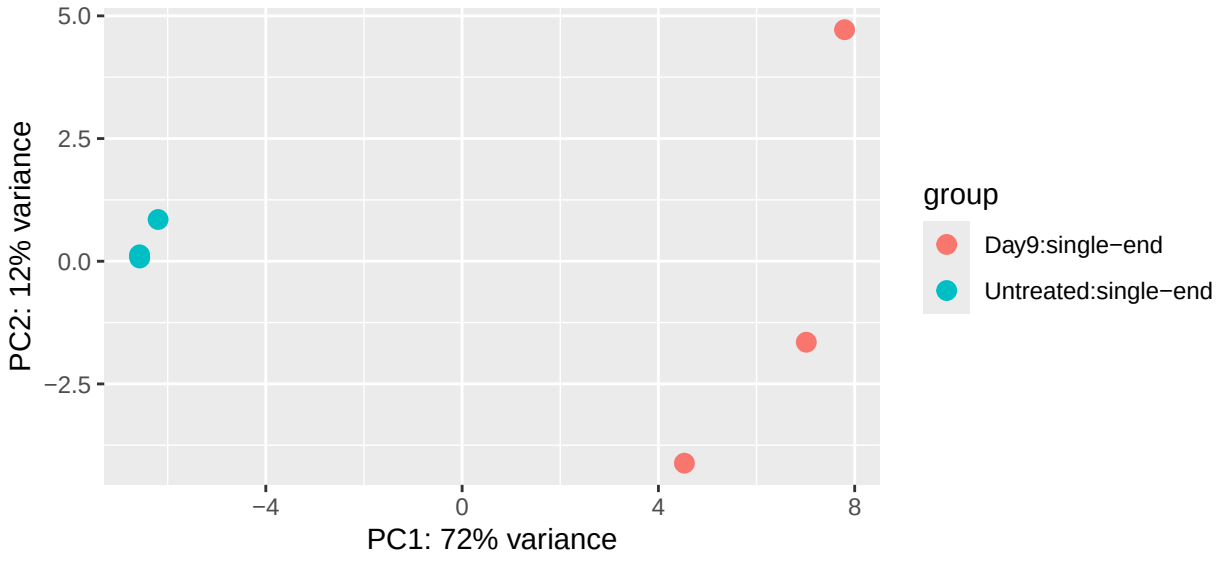

```
# plot PCA
plotPCA(vsd_list[[1]], intgroup=c("condition", "read_type"))
```

```
## using ntop=500 top features by variance
```



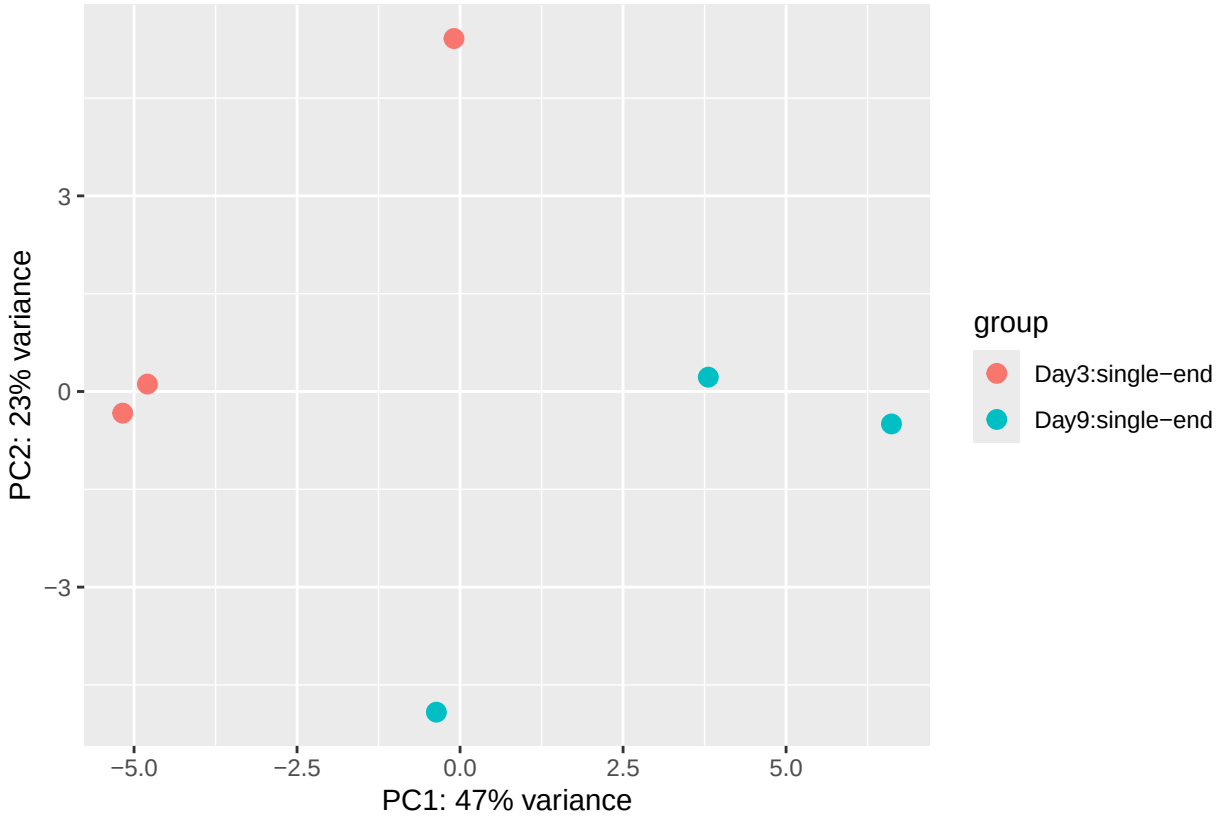

```
plotPCA(vsd_list[[2]], intgroup=c("condition", "read_type"))
```

```
## using ntop=500 top features by variance
```




```
plotPCA(vsd_list[[3]], intgroup=c("condition", "read_type"))
```

```
## using ntop=500 top features by variance
```




```
# customize your PCA plot with ggplot2
for(i in 1:length(results_list)){
  pcaData <- plotPCA(vsd_list[[i]], intgroup=c("condition", "read_type"), returnData=TRUE)
  percentVar <- round(100 * attr(pcaData, "percentVar"))
  ggplot(pcaData, aes(PC1, PC2, color=condition, shape=type)) +
    geom_point(size=3) +
```

```
    xlab(paste0("PC1: ",percentVar[1],"% variance")) +
    ylab(paste0("PC2: ",percentVar[2],"% variance")) +
    coord_fixed()
}
```

```
## using ntop=500 top features by variance
## using ntop=500 top features by variance
## using ntop=500 top features by variance
```