Lab journal

- Wouter Zeevat
- Thema 7

Chapter 3

3.1 Loading data into R

The data consists of 9 groups, a Deletion, Duplication and control group. And for each of those IPSC, 1m and 3m.

```
library(GEOquery)
```

```
## Loading required package: Biobase

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

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

## Setting options('download.file.method.GEOquery'='auto')

## Setting options('GEOquery.inmemory.gpl'=FALSE)
```

```
counts <- read.table('GSE142174_16p11.2_CNVs_autism_organoids_study_counts.csv', sep=',', header=TRUE, 
metadata <- getGEO(filename = 'GSE142174_series_matrix.txt')
```

```
## Rows: 0 Columns: 109

## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (109): ID_REF, GSM4222011, GSM4222012, GSM4222013, GSM4222014, GSM422201...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

## File stored at:

## /tmp/RtmpcfPb75/GPL20301.soft
```

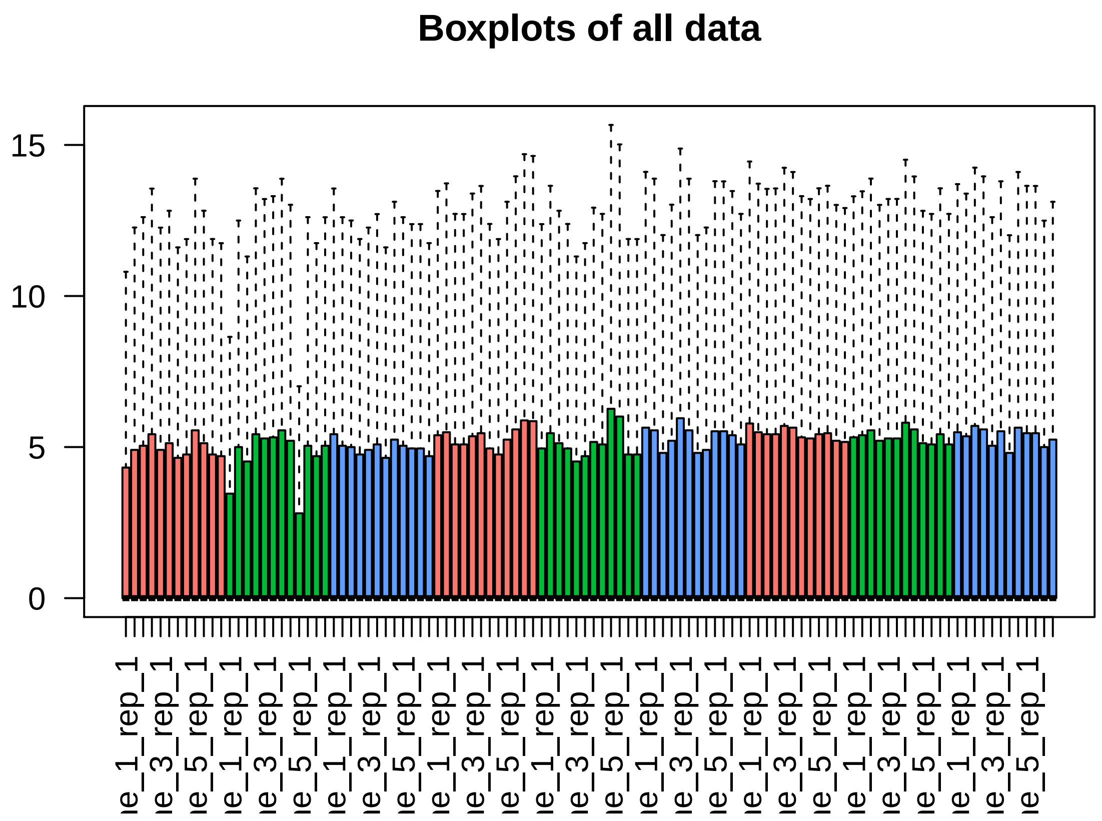
```
names(counts) <- metadata@phenoData@data$title
ipsc_control <- 1:12
ipsc_del <- 13:24
ipsc_dup <- 25:36
one_m_control <- 37:48
one_m_del <- 49:60
one_m_dup <- 61:72
three_m_control <- 73:84
three_m_del <- 85:96
three_m_dup <- 97:108
```

3.2 Example data

3.3 Visualizing using boxplot and density plot

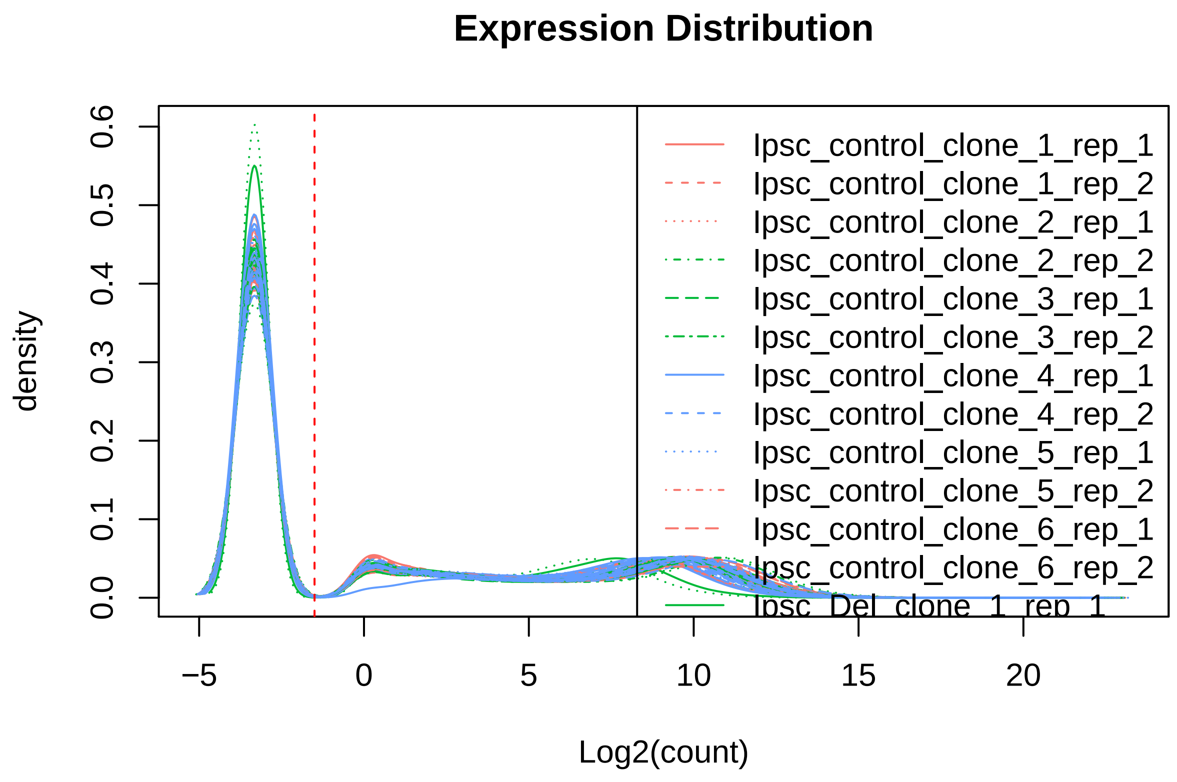
```
library(pander)
```

```
library(scales)
myColors <- hue_pal()(3)
boxplot(log2(counts + 1), outline=FALSE, las=2, col=rep(myColors, each=12), main='Boxplots of all data'
```



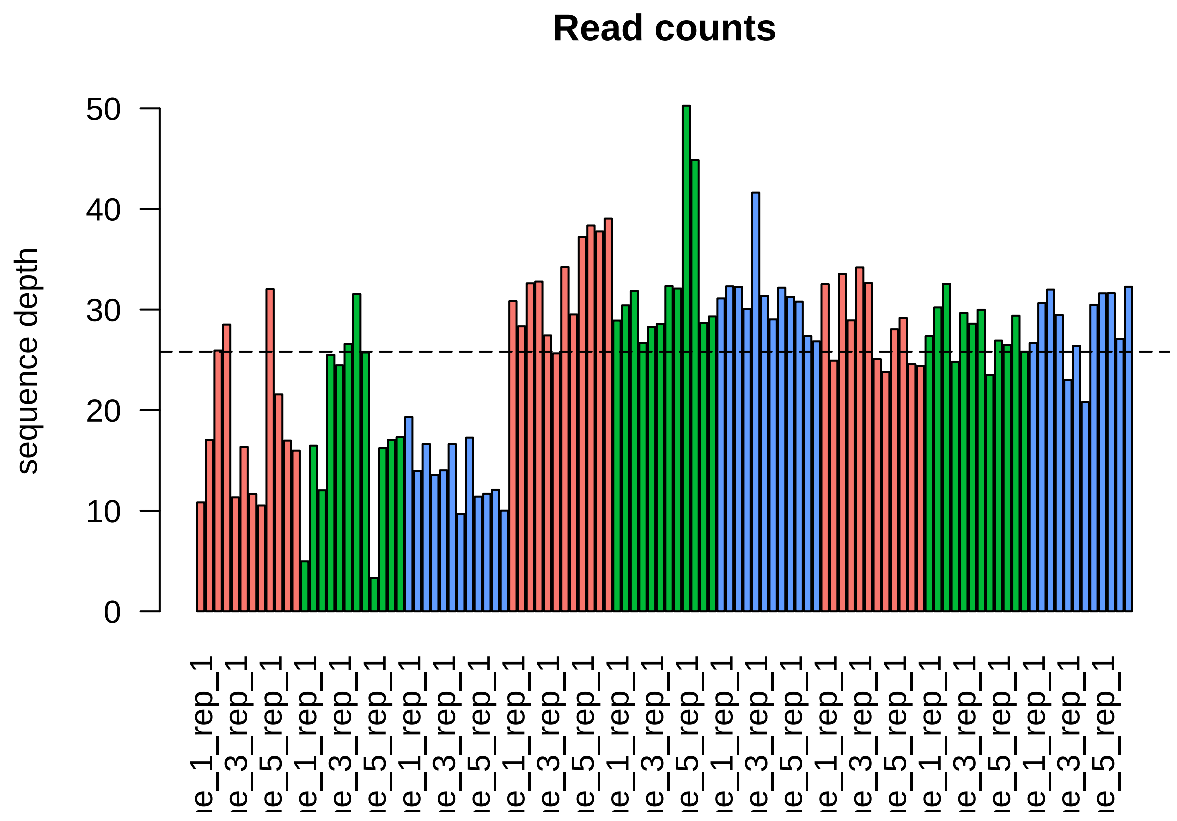
This boxplot shows all values, there are not a lot of patterns to see but it looks like the data is not corrupt.

```
library(affy)

myColors <- hue_pal()(3)

## Plot the log2-transformed data with a 0.1 pseudocount
plotDensity(log2(counts + 0.1), col=rep(myColors, each=12),
            lty=c(1:ncol(counts)), xlab='Log2(count)',
            main='Expression Distribution')

## Add a legend and vertical line
legend('topright', names(counts), lty=c(1:ncol(counts)),
       col=rep(myColors, each=3))
abline(v=-1.5, lwd=1, col='red', lty=2)
```



This graph shows the expression distribution. There's a few ones sticking out but its mostly normal.

```
barplot(colSums(counts) / 1e6, col=rep(myColors, each=12), las=2, main='Read counts', ylab='sequence dep
abline(h = mean(colSums(counts) / 1e6), col = "Black", lty = 5)
```



###

3.4 Sample distance using a heatmap

```r
library(pheatmap)
library(DESeq2)
```

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
##
## 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: DelayedArray
```

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

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

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

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

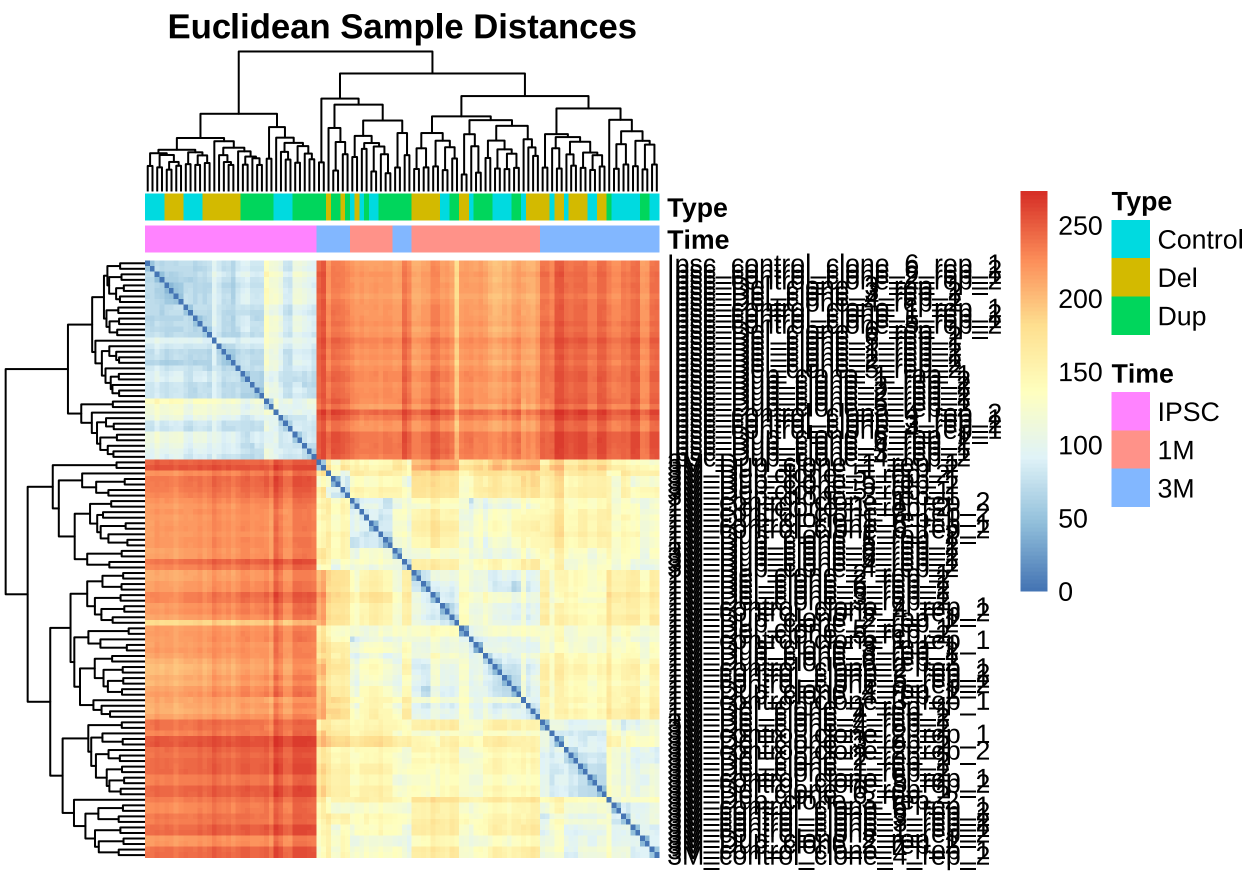
```
## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges
```

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

```r
(ddsMat <- DESeqDataSetFromMatrix(countData = counts,
                                  colData = data.frame(samples = names(counts)),
                                  design = ~ 1))
```

```
## class: DESeqDataSet
## dim: 57820 108
## metadata(1): version
## assays(1): counts
## rownames(57820): ENSG00000000003 ENSG00000000005 ... ENSGR0000266731
##   ENSGR0000270726
## rowData names(0):
## colnames(108): Ipsc_control_clone_1_rep_1 Ipsc_control_clone_1_rep_2
##   ... 3M_Dup_clone_6_rep_1 3M_Dup_clone_6_rep_2
## colData names(1): samples
```

```
rld.dds <- vst(ddsMat)
# 'Extract' normalized values
rld <- assay(rld.dds)
sampledists <- dist( t( rld ))

# Convert the 'dist' object into a matrix for creating a heatmap
sampleDistMatrix <- as.matrix(sampledists)

# The annotation is an extra layer that will be plotted above the heatmap columns
annotation <- data.frame(Time = factor(rep(1:3, each = 36),
                                       labels = c("IPSC", "1M", "3M")),
                         Type = factor(rep(rep(1:3, each = 12), 3),
                                       labels = c("Control", "Del", "Dup")))

# Set the rownames of the annotation dataframe  to the sample names (required)
rownames(annotation) <- names(counts)

pheatmap(sampleDistMatrix, show_colnames = FALSE,
         annotation_col = annotation,
         clustering_distance_rows = sampledists,
         clustering_distance_cols = sampledists,
         main = "Euclidean Sample Distances")
```



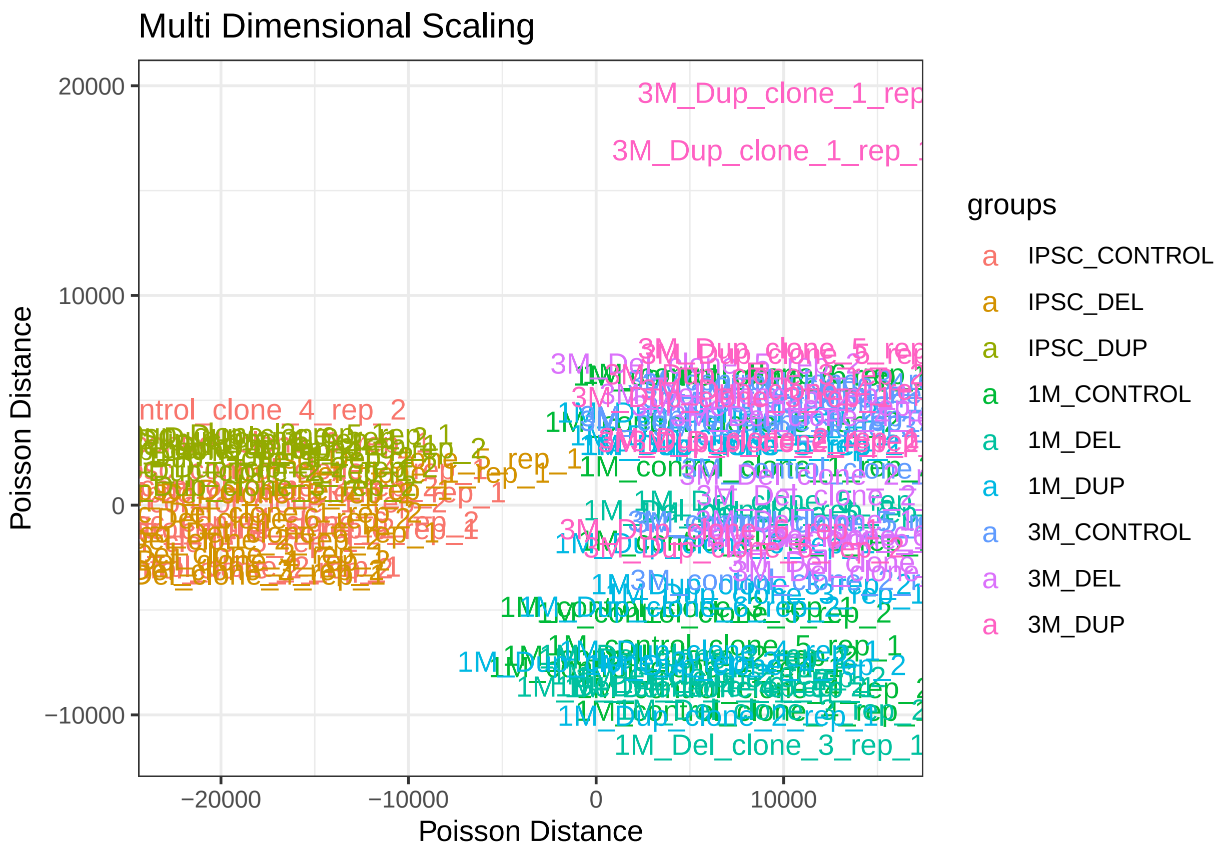
Like in the example, the samples are clustered together pretty well.

3.4.4 Multi-Dimensional Scaling

```
library('PoiClaClu')
# Note: uses the raw-count data, PoissonDistance performs normalization
# set by the 'type' parameter (uses DESeq)
dds <- assay(ddsMat)
poisd <- PoissonDistance( t(dds), type = "deseq")
# Extract the matrix with distances
samplePoisDistMatrix <- as.matrix(poisd$dd)
# Calculate the MDS and get the X- and Y-coordinates
mdsPoisData <- data.frame( cmdscale(samplePoisDistMatrix) )

# And set some better readable names for the columns
names(mdsPoisData) <- c('x_coord', 'y_coord')


# Separate the annotation factor (as the variable name is used as label)
groups <- factor(rep(1:9, each=12),
                 labels = c("IPSC_CONTROL", "IPSC_DEL", "IPSC_DUP", "1M_CONTROL", "1M_DEL", "1M_DUP", "3
coldata <- names(counts)
```

```
# Create the plot using ggplot
library(ggplot2)
```

```
ggplot(mdsPoisData, aes(x_coord, y_coord, color = groups, label = coldata)) +
  geom_text(size = 4) +
  ggtitle('Multi Dimensional Scaling') +
  labs(x = "Poisson Distance", y = "Poisson Distance") +
  theme_bw()
```



This graph shows the outliers very well. In this case, there aren't many that differ from the others too much. This is why i decided to not remove them as they can still be crusial to the result.

Chapter 4

4.1 Preprocessing

```
# Perform a naive FPM normalization
# Note: log transformation includes a pseudocount of 1
counts.fpm <- log2( (counts / (colSums(counts) / 1e6)) + 1 )
```

The data will now be normalized by calculating the FPM.