# Pratica RNAseq

```
library(DESeq2)
library(ggplot2)
library(tximeta)
library(pheatmap)
library(viridis)
setwd("~/PRATICA_RNASEQ/")
```

Load the information about the experiment, sample names, and condition

```
targets<-read.table("metadata.csv",
                    header=T,
                    sep = ",",
                    stringsAsFactors=FALSE,
                    col.names = c("names", "condition", "cell"))
targets$names<-as.factor(targets$names)
targets$cell<-as.factor(targets$cell)
targets$group <- as.factor(paste0(targets$condition, "_", targets$cell))
```

make a list of the quantification files created by salmon. Please note these are only strings

```
files<-file.path(paste0("Quantification/",targets$names,"/quant.sf",se =""))
```

check that the string (file names) created in the previous steps are actual files on disk

```
file.exists(files)
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Add the file names to the description of experiment.

```
targets$files<-files
row.names(targets)<-targets$names
```

#check the targets object

```
targets
```

```
##                   names     condition    cell                 group
## SRR1039508 SRR1039508     Untreated  N61311      Untreated_N61311
## SRR1039509 SRR1039509 Dexamethasone  N61311  Dexamethasone_N61311
## SRR1039512 SRR1039512     Untreated N052611     Untreated_N052611
## SRR1039513 SRR1039513 Dexamethasone N052611 Dexamethasone_N052611
## SRR1039516 SRR1039516     Untreated N080611     Untreated_N080611
## SRR1039517 SRR1039517 Dexamethasone N080611 Dexamethasone_N080611
## SRR1039520 SRR1039520     Untreated N061011     Untreated_N061011
## SRR1039521 SRR1039521 Dexamethasone N061011 Dexamethasone_N061011
##                                          files
## SRR1039508 Quantification/SRR1039508/quant.sf
## SRR1039509 Quantification/SRR1039509/quant.sf
## SRR1039512 Quantification/SRR1039512/quant.sf
## SRR1039513 Quantification/SRR1039513/quant.sf
## SRR1039516 Quantification/SRR1039516/quant.sf
## SRR1039517 Quantification/SRR1039517/quant.sf
## SRR1039520 Quantification/SRR1039520/quant.sf
## SRR1039521 Quantification/SRR1039521/quant.sf
```

We will import the transcript quantification generated by salmon, and the experiment description using tximeta Check the manual of tximeta for further details: https://bioconductor.org/packages/3.14/bioc/vignettes/tximeta/inst/doc/tximeta.html

Now, before loading our data, create tthe transcriptome object, this will help to carry out several operations, like summarizing the transcript expression levels at the gene level, get functional information about the genes, and so on.

```
makeLinkedTxome(indexDir="salmon_index",
                source="LocalEnsembl",
                organism="Homo sapiens",
                release="38",
                genome="GRCh38.p13",
                fasta="./references/gencode.v38.transcripts.fa.gz",
                gtf="./references/gencode.v38.annotation.gtf.gz",
                write=FALSE)
```

Now create the tximeta object that has the salmon data, the experiment description, and access to the transcriptome information. This object is called SummarizedExperiment

```
se <- tximeta(targets)
```

Check the size of the object

```
dim(se)
```

```
## [1] 236186      8
```

check the names of the transcripts, confirm that you are seeing the transcript names and not gene names. How woudl you now.

```
head(rownames(se))
```

```
## [1] "ENST00000456328.2" "ENST00000450305.2" "ENST00000488147.1"
## [4] "ENST00000619216.1" "ENST00000473358.1" "ENST00000469289.1"
```

Nos exploit the transcriptome information to summarize the expression data at the gene level. Do you understand what this means?

```r
gse <- summarizeToGene(se)
```

Check the size of the object and check the names of the genes. Should be now gene names and not transcripts names

```r
dim(gse)
```

```
## [1] 60230     8
```

```r
head(rownames(gse))
```

```
## [1] "ENSG00000000003.15" "ENSG00000000005.6"  "ENSG00000000419.14"
## [4] "ENSG00000000457.14" "ENSG00000000460.17" "ENSG00000000938.13"
```

these two object are compossed of colData, ranges and assay data, see figure let's see each of these elements in our objects se and gse See the figure in section 2.5 of https://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html

```r
colData(gse)
```

```
## DataFrame with 8 rows and 4 columns
##                 names     condition     cell                 group
##              <factor>   <character> <factor>              <factor>
## SRR1039508 SRR1039508     Untreated  N61311  Untreated_N61311
## SRR1039509 SRR1039509 Dexamethasone  N61311  Dexamethasone_N61311
## SRR1039512 SRR1039512     Untreated  N052611 Untreated_N052611
## SRR1039513 SRR1039513 Dexamethasone  N052611 Dexamethasone_N052611
## SRR1039516 SRR1039516     Untreated  N080611 Untreated_N080611
## SRR1039517 SRR1039517 Dexamethasone  N080611 Dexamethasone_N080611
## SRR1039520 SRR1039520     Untreated  N061011 Untreated_N061011
## SRR1039521 SRR1039521 Dexamethasone  N061011 Dexamethasone_N061011
```

```r
head(assay(gse))
```

```
##                    SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003.15     56.109     35.919     61.439     46.000     77.000
## ENSG00000000005.6       0.000      0.000      0.000      0.000      0.000
## ENSG00000000419.14     39.002     54.013     39.972     41.000     46.001
## ENSG00000000457.14     30.647     29.552     22.995     25.502     23.330
## ENSG00000000460.17      5.700      9.453      2.000      3.499     11.000
## ENSG00000000938.13      0.000      0.000      0.000      0.000      0.000
##                    SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003.15     77.984     63.480     55.973
## ENSG00000000005.6       0.000      0.000      0.000
## ENSG00000000419.14     41.019     40.390     36.000
## ENSG00000000457.14     22.254     19.409     27.784
## ENSG00000000460.17     11.737     10.591      5.000
## ENSG00000000938.13      0.000      0.000      0.000
```

```
rowRanges(gse)
```

```
## GRanges object with 60230 ranges and 2 metadata columns:
##                      seqnames              ranges strand |            gene_id
##                         <Rle>           <IRanges>  <Rle> |        <character>
##   ENSG00000000003.15     chrX 100627108-100639991      - | ENSG00000000003.15
##    ENSG00000000005.6     chrX 100584936-100599885      + |  ENSG00000000005.6
##   ENSG00000000419.14    chr20   50934867-50959140      - | ENSG00000000419.14
##   ENSG00000000457.14     chr1 169849631-169894267      - | ENSG00000000457.14
##   ENSG00000000460.17     chr1 169662007-169854080      + | ENSG00000000460.17
##                  ...      ...                 ...    ... .                ...
##    ENSG00000288721.1     chr6   41793314-41921139      - |  ENSG00000288721.1
##    ENSG00000288722.1     chrX 154886355-154888061      + |  ENSG00000288722.1
##    ENSG00000288723.1     chr1 241722926-241848128      - |  ENSG00000288723.1
##    ENSG00000288724.1     chr3   46284775-46293795      - |  ENSG00000288724.1
##    ENSG00000288725.1    chr16   56430556-56501497      - |  ENSG00000288725.1
##                                                                        tx_ids
##                                                               <CharacterList>
##   ENSG00000000003.15  ENST00000373020.9,ENST00000612152.4,ENST00000614008.4,...
##    ENSG00000000005.6                      ENST00000373031.5,ENST00000485971.1
##   ENSG00000000419.14 ENST00000466152.5,ENST00000371582.8,ENST00000371588.10,...
##   ENSG00000000457.14 ENST00000367771.11,ENST00000367770.5,ENST00000367772.8,...
##   ENSG00000000460.17  ENST00000498289.5,ENST00000472795.5,ENST00000359326.9,...
##                  ...                                                      ...
##    ENSG00000288721.1      ENST00000684631.1,ENST00000682596.1,ENST00000683313.1
##    ENSG00000288722.1                                        ENST00000610495.2
##    ENSG00000288723.1                                        ENST00000684005.1
##    ENSG00000288724.1                                        ENST00000683399.1
##    ENSG00000288725.1                                        ENST00000684388.1
##   -------
##   seqinfo: 25 sequences (1 circular) from an unspecified genome; no seqlengths
```

With these object (gse) we can now start the differential gene expression analysis For that there are many different R packages that could be used today we are going to use the DESeq2 package. So, first we need to transform our SummarizedExperiment (gse) object to an object that is native to DESeq2, which is DESeqDataSet, in addition to the data, the DESeqDataSet requires a description of the experimental design, in which we tell what are the factor of interest in the study, and/or the factors that should be sources of variation and how to deal with them. This is done using a formula, with the same sintax as in simple linear models (lm) in R.

In this particular example, we have at least (that we know of) sources of variation. First, the condition, samples treated or not with Dexamethasone. Second, the cell line. The primary ASM cell were obtained from four donors, and then divided in the treated and untreated groups. So we can have an effect of the donor. Check you targets object to see this. We will use these two factors in our analyses. We want to test the effect of Dexamethasone, while controlling for the effect of different cellLines. Note that this is a paired experimental design. As each sample (cellLine) was treated and not treated in that case we will specify the formula as ~ cellLine + condition

```
dds <- DESeqDataSet(gse, design = ~condition+cell)
```

It is common that genes in the dataset at not expressed at all, and others to be very lowly expressed, it is a good practice to remove these from further analyses

```
nrow(dds) #Number of genes pre-filtering
```

```
## [1] 60230
```

```
keep <- rowSums(counts(dds)) > 1
dds <- dds[keep,]
nrow(dds) #Number of genes after filtering out non-expressed genes.
```

```
## [1] 22636
```

Check the effect of normalization

```
head(counts(dds),2)
```

```
##                      SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003.15         56         36         61         46         77
## ENSG00000000419.14         39         54         40         41         46
##                      SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003.15         78         63         56
## ENSG00000000419.14         41         40         36
```

```
dds <- estimateSizeFactors(dds)
head(counts(dds,normalized=TRUE),2)
```

```
##                      SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003.15   44.46754   29.94703   81.88886   40.44441   67.13792
## ENSG00000000419.14   38.29760   58.25576   38.58919   42.42453   42.50157
##                      SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003.15   89.51814   93.20195   48.51856
## ENSG00000000419.14   40.72133   37.65602   39.40506
```

```
colSums(assay(gse))
```

```
## SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517 SRR1039520
##    1788789    1752398    1711969    1661466    1638343    1636627    1652720
## SRR1039521
##    1645795
```

It is also a good idea to remove genes in which a given number of samples do not have an count

```
keep <- rowSums(counts(dds) >= 5) >= 4
dds <- dds[keep,]
nrow(dds) # Number of genes, that have counts in at least 4 samples, and the sum of counts is greated t
```

```
## [1] 12496
```

Now that the data is in the proper format and filtered we can carry out some exploratory analyses to check whether the data looks good

Many os the exploratory analysis require homocedastic data, which our counts are not. In DESeq2 the function vst and rlog, can normalize the data to make it more homocedastic. As a rule of thum vst is better when you have more than 30 samples

```r
rld <- rlog(dds, blind = TRUE)
head(assay(rld))
```

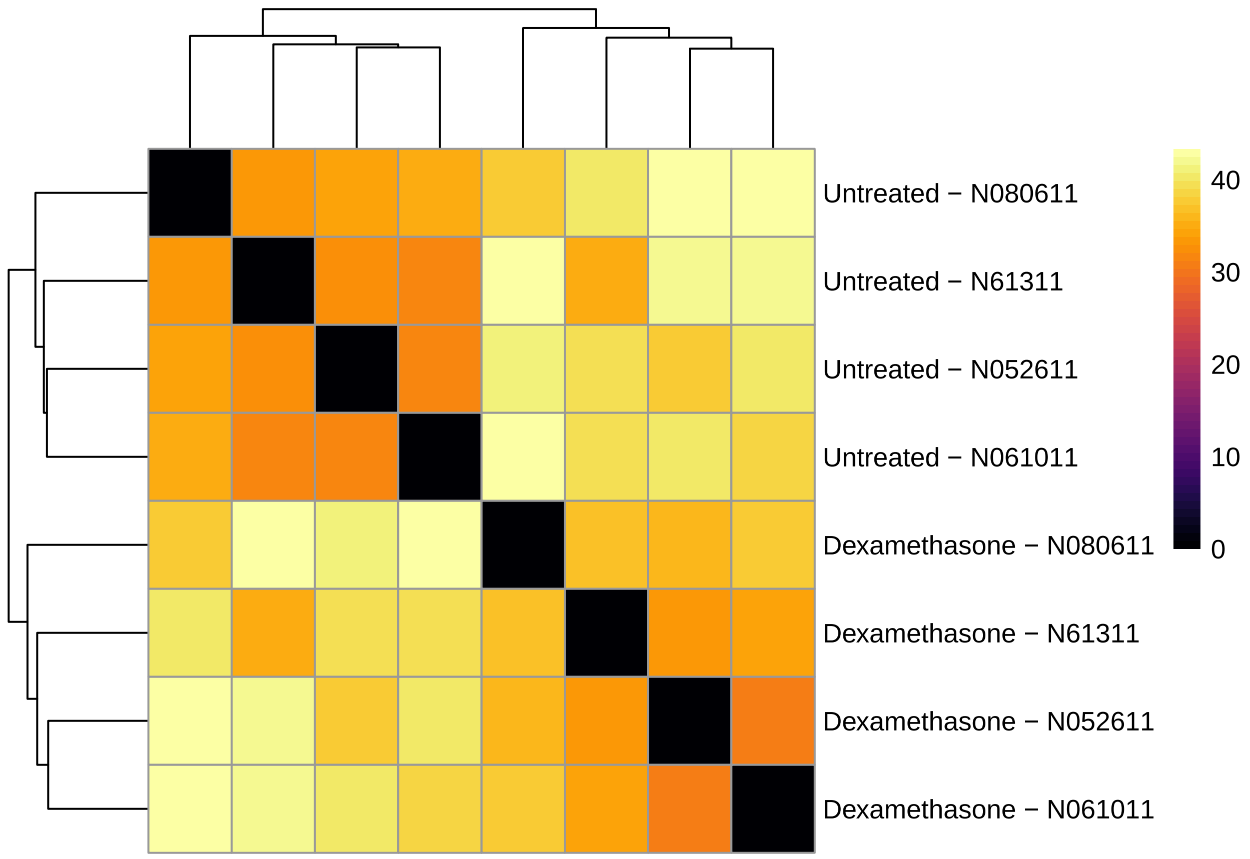
```
##                      SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003.15     5.669486   5.411940   6.132500   5.605813   5.978105
## ENSG00000000419.14     5.329070   5.627498   5.334025   5.398322   5.399654
## ENSG00000000457.14     4.661080   4.683958   4.567897   4.858966   4.463062
## ENSG00000000460.17     2.710707   3.191735   2.643562   2.710029   3.287850
## ENSG00000000971.16     8.355633   8.681279   8.717682   9.023315   8.728717
## ENSG00000001036.14     7.378607   7.354500   7.537588   7.311982   7.250493
##                      SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003.15     6.209172   6.237266   5.732569
## ENSG00000000419.14     5.370325   5.317605   5.348540
## ENSG00000000457.14     4.561131   4.576623   4.843604
## ENSG00000000460.17     3.008316   2.910836   2.707125
## ENSG00000000971.16     9.153701   8.788389   9.288140
## ENSG00000001036.14     7.075827   7.401430   7.059146
```

Now let's look at the similarity among samples we will compute the euclidean distance between the samples with the function dist This function assumes that the samples are the rows, so we must first transpose the matrix We will visualize the distances between the sampleas as a heatmap

```r
sampleDists <- dist(t(assay(rld)))
sampleDists
```

```
##            SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517
## SRR1039509   35.48345
## SRR1039512   32.42730   39.11528
## SRR1039513   41.89743   33.74853   37.39973
## SRR1039516   33.69166   40.44933   34.24658   42.86149
## SRR1039517   42.85923   36.91667   40.91284   36.26586   37.64365
## SRR1039520   31.79145   39.64518   31.46308   40.20671   35.03774   42.51003
## SRR1039521   42.11879   34.48375   40.43918   31.07203   43.33696   37.49779
##            SRR1039520
## SRR1039509
## SRR1039512
## SRR1039513
## SRR1039516
## SRR1039517
## SRR1039520
## SRR1039521   38.20044
```

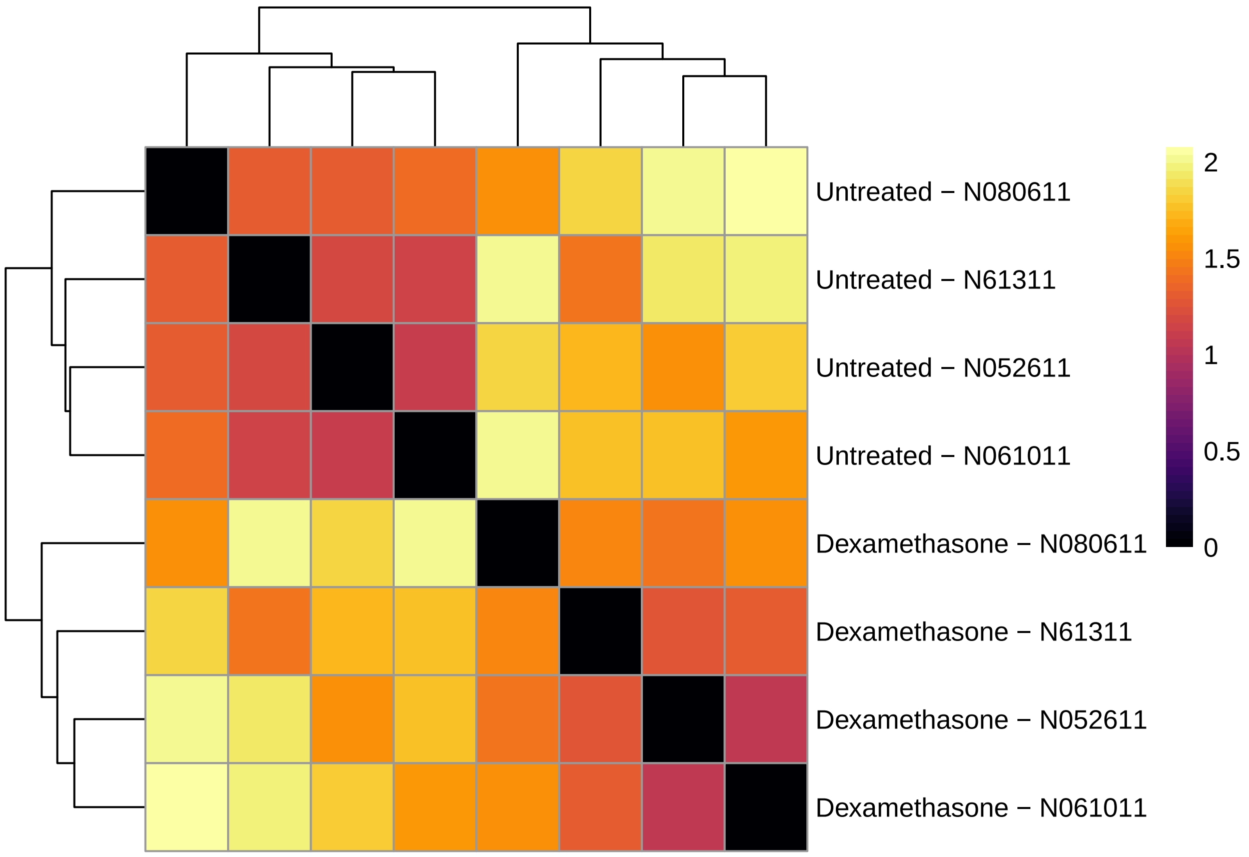
```r
sampleDistMatrix <- as.matrix( sampleDists )
rownames(sampleDistMatrix) <- paste( rld$condition, rld$cell, sep = " - " )
colnames(sampleDistMatrix) <- NULL
colors <- inferno(50)
pheatmap(sampleDistMatrix,
         clustering_distance_rows = sampleDists,
         clustering_distance_cols = sampleDists,
         col = colors)
```



```r
sampleDists <- dist(t(assay(rld)))
sampleDists
```

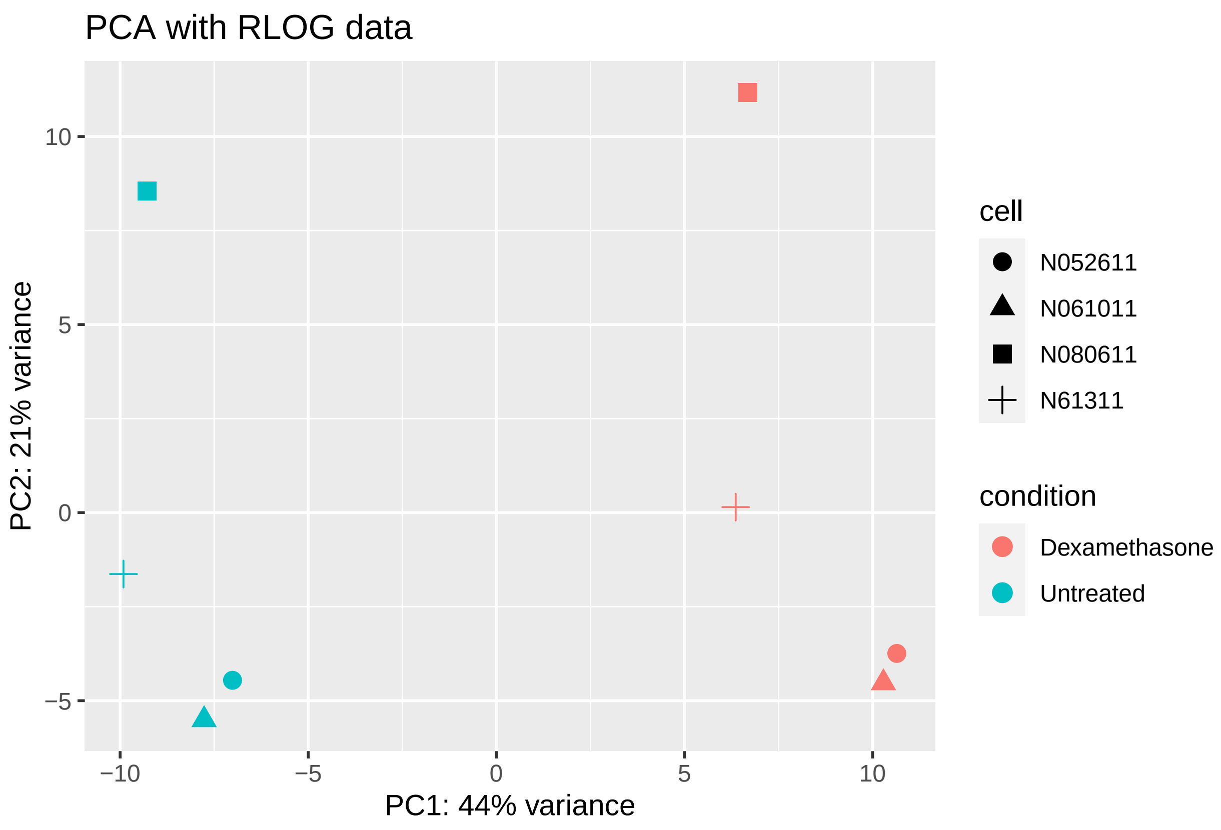
```
##            SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517
## SRR1039509   35.48345
## SRR1039512   32.42730   39.11528
## SRR1039513   41.89743   33.74853   37.39973
## SRR1039516   33.69166   40.44933   34.24658   42.86149
## SRR1039517   42.85923   36.91667   40.91284   36.26586   37.64365
## SRR1039520   31.79145   39.64518   31.46308   40.20671   35.03774   42.51003
## SRR1039521   42.11879   34.48375   40.43918   31.07203   43.33696   37.49779
##            SRR1039520
## SRR1039509
## SRR1039512
## SRR1039513
## SRR1039516
## SRR1039517
## SRR1039520
## SRR1039521   38.20044
```

```r
samplecor <- as.dist((1-(cor(assay(rld), method='pearson')))*100)
sampleCorMatrix <- as.matrix( samplecor )
rownames(sampleCorMatrix) <- paste( rld$condition, rld$cell, sep = " - " )
colnames(sampleCorMatrix) <- NULL
pheatmap(sampleCorMatrix,
         clustering_distance_rows = samplecor,
         clustering_distance_cols = samplecor,
         col=colors)
```



It is also common to show a PCA plot of the samples, colored with the factors of interest

```
pcaData <- plotPCA(rld, intgroup = c( "condition",
                                      "cell"),
                   returnData = TRUE)
percentVar <- round(100 * attr(pcaData, "percentVar"))
ggplot(pcaData, aes(x = PC1, y = PC2, color = condition, shape = cell)) +
  geom_point(size =3) +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance")) +
  coord_fixed() +
  ggtitle("PCA with RLOG data")
```


PCA with RLOG data
PC2: 21% variance
PC1: 44% variance
10
5
0
−5
−10
−5
0
5
10
cell
N052611
N061011
N080611
N61311
condition
Dexamethasone
Untreated