

Gene Expression Practical

Igor Ruiz de los Mozos PhD

Universidad de Navarra

Máster en Métodos Computacionales en Ciencias

Análisis e interpretación de datos de alto rendimiento

Introduction

Students will be handed real human RNA-seq data (Himes et al 2014) to investigate differential gene expression (DGE) under different experimental conditions.

Characterization transcriptomic changes in four primary human ASM cell lines that were treated with dexamethasone — a potent synthetic glucocorticoid ($1 \text{ } \mu\text{M}$ for 18 hours). R and Bioconductor statistical packages to research variation at gene level. R studio NGS visualization technologies will be used to display gene expression differences and gene ontology to infer biological meaning of their findings.

Additional Reading and Credits

This practice material was adapted from Michael Love, DeSeq2 tutorial 2018 (kindly authorized us to reproduce). RNA-seq workflow: gene-level exploratory analysis and differential expression. 2018 **

<https://www.bioconductor.org/packages/devel/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html>

Introduction to DGE. Teaching materials at the Harvard Chan Bioinformatics Core *These are open access materials distributed under the terms of the Creative Commons Attribution license (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.* https://hbctraining.github.io/DGE_workshop/lessons/04_DGE_DESeq2_analysis.html

Introduction to differential gene expression analysis using RNA-seq. Friederike Dündar, Luce Skrabanek, Paul Zumbo 2015 * <http://chagall.med.cornell.edu/RNASeqcourse/Intro2RNASEq.pdf>

Interesting paper describing all the steps in a RNA-seq DGE experiment ** Conesa A, Madrigal P, Tarazona S, et al. A survey of best practices for RNA-seq data analysis. Genome Biology. 2016;17:13. doi:10.1186/s13059-016-0881-8.

Love, Michael I., et al. "RNA-Seq workflow: gene-level exploratory analysis and differential expression." F1000Research 4 (2015). *

List of RNA-Seq bioinformatics tools Extensive list of RNA-seq bioinformatic tools.

https://en.wikipedia.org/wiki/List_of_RNA-Seq_bioinformatics_tools

Differential gene expression using R and Bioconductor packages.

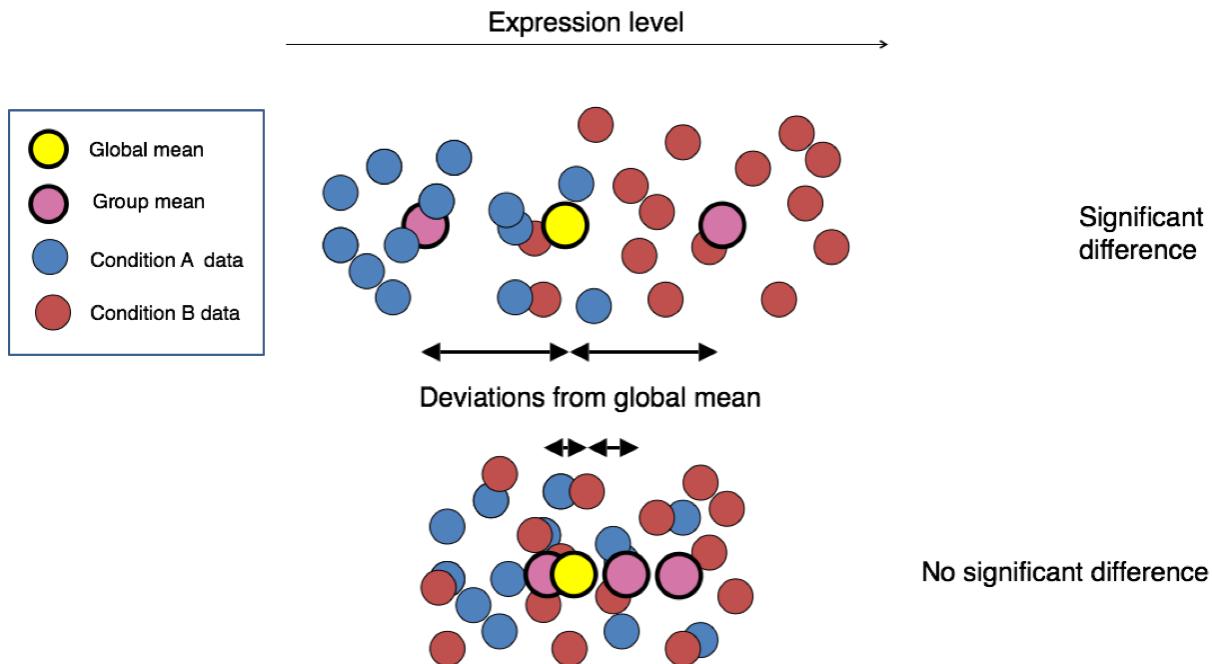
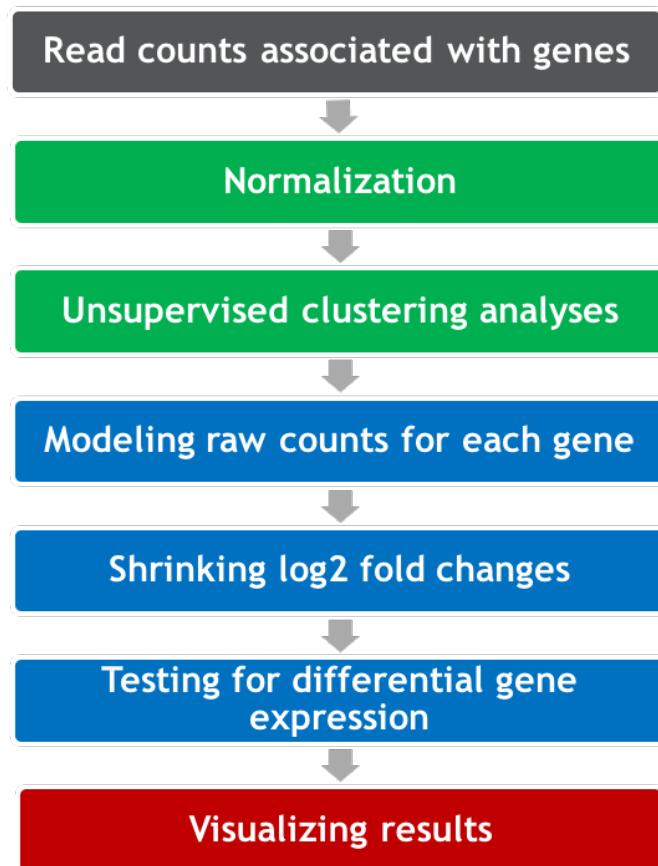


Figure 1: mean normalization



DGE bioinformatic schema

```
# # Download BiocManager
# if (!requireNamespace("BiocManager", quietly = TRUE))
#   install.packages("BiocManager")
# # Download airway library
# BiocManager::install("airway")

## For old R distribution <4.0
# source("https://bioconductor.org/biocLite.R")
# #biocLite("airway")
# if (!require(airway)) {
#   install.packages("airway")
#   require(airway)
# }

# Load airway
library("airway")

# Install and load SummarizedExperiment
# BiocManager::install("SummarizedExperiment")
library("SummarizedExperiment")

# Check airway vignette
# vignette("airway")

# Load airway data
data("airway")

# Assign airway data to summarized experiment R object
se <- airway

# Set directory
# WARNING, change to your system path!!!

# # Load directory from MAC
# directory <- "/Users/mozosi/Dropbox (UCL-MN Team)/Class/Unav2021/ML/DGE"

# Load directory from Windows
# directory <- "C:/Users/d682011.ADMON-CFNAVARRA/OneDrive - NAVARRA SOCIEDADES PÚBLICAS/Others/classes/"
directory <- "/home/rstudio"

setwd(directory)
list.files()
```

Airway data load

```
## [1] "DESeq2_Sig_results_dex_Vs_untrt.tab"
## [2] "Dockerfile_rbase"
## [3] "Dockerfile_rstudio"
```

```

## [4] "Dockerfile_tidyverse_verse"
## [5] "Dockerfile_tidyverse_verse_dockering"
## [6] "Dockerfile_tidyverse_works"
## [7] "Dockerfile_verse"
## [8] "figures"
## [9] "Gene_expression_practical_UNAV_cache"
## [10] "Gene_expression_practical_UNAV_files"
## [11] "Gene_expression_practical_UNAV.html"
## [12] "Gene_expression_practical_UNAV.log"
## [13] "Gene_expression_practical_UNAV.Rmd"
## [14] "Gene_expression_practical_UNAV.tex"
## [15] "LICENSE"
## [16] "Merge_GO_Gene_Set_Enrichment_Analysis_CC_.pdf"
## [17] "README.md"
## [18] "se_data.rds"
## [19] "Significant_DE_dex_trt_vs_untrt_metadata.tab"

# Save an object to a file
saveRDS(se, file = "se_data.rds")

# Restore the object. In case you will need to restore the data object
# readRDS(file = "se_data.rds")

```

Summarized experiment.

The component parts of a SummarizedExperiment object. The assay (pink block) contains the matrix of counts, the rowRanges (blue block) contains information about the genomic ranges and the colData (green block) contains information about the samples. The highlighted line in each block represents the first row (note that the first row of colData lines up with the first column of the assay).

```

# Total sum of read counts
colSums(assay(se))

## SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517 SRR1039520
##   20637971    18809481    25348649    15163415    24448408    30818215    19126151
## SRR1039521
##   21164133

# Head of summarizeexperiment object
head(assay(se), 3)

```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516	SRR1039517
## ENSG00000000003	679	448	873	408	1138	
## ENSG00000000005	0	0	0	0	0	
## ENSG00000000419	467	515	621	365	587	
	SRR1039517	SRR1039520	SRR1039521			
## ENSG00000000003	1047	770	572			
## ENSG00000000005	0	0	0			
## ENSG00000000419	799	417	508			

```

# Gene genomic ranges with genomic ranges
rowRanges(se)

```

```

## GRangesList object of length 63677:
## $ENSG00000000003
## GRanges object with 17 ranges and 2 metadata columns:

```



Figure 3: Summarized experiment structure

```

##      seqnames      ranges strand |  exon_id      exon_name
##      <Rle>      <IRanges>  <Rle> | <integer>      <character>
## [1]     X 99883667-99884983     - | 667145 ENSE00001459322
## [2]     X 99885756-99885863     - | 667146 ENSE00000868868
## [3]     X 99887482-99887565     - | 667147 ENSE00000401072
## [4]     X 99887538-99887565     - | 667148 ENSE00001849132
## [5]     X 99888402-99888536     - | 667149 ENSE00003554016
## ...
## [13]    ...       ...       ... | ...
## [14]    X 99890555-99890743     - | 667156 ENSE00003512331
## [15]    X 99891188-99891686     - | 667158 ENSE00001886883
## [16]    X 99891605-99891803     - | 667159 ENSE00001855382
## [17]    X 99891790-99892101     - | 667160 ENSE00001863395
## [18]    X 99894942-99894988     - | 667161 ENSE00001828996
##
##      -----
##  seqinfo: 722 sequences (1 circular) from an unspecified genome
##
## ...
## <63676 more elements>

# Metadata contained in the ranges
str(metadata(rowRanges(se)))

## List of 1
## $ genomeInfo:List of 20
##   ..$ Db type                      : chr "TranscriptDb"
##   ..$ Supporting package            : chr "GenomicFeatures"
##   ..$ Data source                  : chr "BioMart"
##   ..$ Organism                     : chr "Homo sapiens"
##   ..$ Resource URL                 : chr "www.biomart.org:80"
##   ..$ BioMart database              : chr "ensembl"
##   ..$ BioMart database version     : chr "ENSEMBL GENES 75 (SANGER UK)"
##   ..$ BioMart dataset               : chr "hsapiens_gene_ensembl"
##   ..$ BioMart dataset description  : chr "Homo sapiens genes (GRCh37.p13)"
##   ..$ BioMart dataset version      : chr "GRCh37.p13"
##   ..$ Full dataset                 : chr "yes"
##   ..$ miRBase build ID            : chr NA
##   ..$ transcript_nrow              : chr "215647"
##   ..$ exon_nrow                    : chr "745593"
##   ..$ cds_nrow                     : chr "537555"
##   ..$ Db created by                : chr "GenomicFeatures package from Bioconductor"
##   ..$ Creation time                : chr "2014-07-10 14:55:55 -0400 (Thu, 10 Jul 2014)"
##   ..$ GenomicFeatures version at creation time: chr "1.17.9"
##   ..$ RSQLite version at creation time: chr "0.11.4"
##   ..$ DBSCHEMAVERSION             : chr "1.0"

# Annotation of the experiment
colData(se)

## DataFrame with 8 rows and 9 columns
##   SampleName   cell   dex   albut      Run avgLength
##   <factor> <factor> <factor> <factor> <factor> <integer>
## SRR1039508 GSM1275862 N61311 untrt  untrt SRR1039508     126
## SRR1039509 GSM1275863 N61311   trt  untrt SRR1039509     126
## SRR1039512 GSM1275866 N052611 untrt  untrt SRR1039512     126
## SRR1039513 GSM1275867 N052611   trt  untrt SRR1039513      87
## SRR1039516 GSM1275870 N080611 untrt  untrt SRR1039516     120

```

```

## SRR1039517 GSM1275871 N080611 trt untrt SRR1039517 126
## SRR1039520 GSM1275874 N061011 untrt untrt SRR1039520 101
## SRR1039521 GSM1275875 N061011 trt untrt SRR1039521 98
##           Experiment   Sample   BioSample
##                 <factor> <factor> <factor>
## SRR1039508 SRX384345 SRS508568 SAMN02422669
## SRR1039509 SRX384346 SRS508567 SAMN02422675
## SRR1039512 SRX384349 SRS508571 SAMN02422678
## SRR1039513 SRX384350 SRS508572 SAMN02422670
## SRR1039516 SRX384353 SRS508575 SAMN02422682
## SRR1039517 SRX384354 SRS508576 SAMN02422673
## SRR1039520 SRX384357 SRS508579 SAMN02422683
## SRR1039521 SRX384358 SRS508580 SAMN02422677

# Check object attributes and class
#attributes(se)
class(se)

## [1] "RangedSummarizedExperiment"
## attr(,"package")
## [1] "SummarizedExperiment"

```

Differential gene expression with DeSeq2

DeSeq2 is one of the most widely used DGE packages (Love 2014).

* Don't assume that variance and median are equal – not normally distributed. * Allow global variance and check the variance of a single gene versus the rest and whether the changes are due to chance. * In essence, we are testing if the mean expression are significantly different. The null hypothesis is that there is no systematic difference between the average read count values of the different conditions for a given gene.

```

# Download DeSeq2 if necessary
# if (!require(DESeq2)) {
#   install.packages("DESeq2")
#   require(DESeq2)
# }
# BiocManager::install("DESeq2")
# BiocManager::install("DelayedArray")

```

The DESeq2 model

The DESeq2 model and all the steps taken in the software are described in detail in our publication (Love, Huber, and Anders 2014), and we include the formula and descriptions in this section as well. The differential expression analysis in DESeq2 uses a generalized linear model of the form:

$$K_{ij} \sim NB(\lambda) \mu_{ij} \alpha_i * \\ \lambda \mu_{ij} (s_j q_{ij})^*$$

$$\log_2(q_{ij}) = x_j \beta_i + \epsilon$$

where counts K_{ij} for gene i , sample j are modeled using a negative binomial distribution with fitted mean μ_{ij} and a gene-specific dispersion parameter α_i . The fitted mean is composed of a sample-specific size factor s_j and a parameter q_{ij} proportional to the expected true concentration of fragments for sample j . The coefficients β_i give the log2 fold changes for gene i for each column of the model matrix X . Note that the model can be generalized to use sample- and gene-dependent normalization factors s_{ij}

The dispersion parameter α_i defines the relationship between the variance of the observed count and its mean value. In other words, how far do we expect the observed count will be from the mean value, which depends both on the size factor sj and the covariate-dependent part qij as defined above.

```
Var(Kij)=E[(Kij-/\mu\ij)^2]=/\mu\ij+\alpha\ij\mu\ij*
```

An option in DESeq2 is to provide maximum a posteriori estimates of the log2 fold changes in β_i after incorporating a zero-centered Normal prior (betaPrior). While previously, these moderated, or shrunken, estimates were generated by DESeq or nbinomWaldTest functions, they are now produced by the lfcShrink function. Dispersions are estimated using expected mean values from the maximum likelihood estimate of log2 fold changes, and optimizing the Cox-Reid adjusted profile likelihood, as first implemented for RNA-seq data in edgeR (Cox and Reid 1987,edgeR_GLM). The steps performed by the DESeq function are documented in its manual page DESeq; briefly, they are:

1. estimation of size factors sj by *estimateSizeFactors*
2. estimation of dispersion α_i by *estimateDispersions*
3. negative binomial GLM fitting for β_i and Wald statistics by *nbinomWaldTest*

For access to all the values calculated during these steps, see the link below

<http://bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html#access>

```
# Load DeSeq2
library("DESeq2")

# Order of factors before relevel
se$dex

## [1] untrt trt   untrt trt   untrt trt   untrt trt
## Levels: trt untrt

# When we talk about differentially gene expression (DGE) we are looking for changes in expression on the
se$dex <- relevel(se$dex, "untrt")
# Order of factors after relevel. Untreated came to the first position on the factors levels
se$dex

## [1] untrt trt   untrt trt   untrt trt   untrt trt
## Levels: untrt trt

# Construct a DESeqDataSet data object
dds_raw <- DESeqDataSet(se, design = ~ cell + dex)
```

Note on sample comparison

```
# DO NOT RUN! contrast <- c("condition", "treatment", "control") results(dds, contrast = contrast, alpha = alpha_threshold)
```

```
# Result table should have the control at the end of the comparison log2 fold change (MAP): samplotype treatment vs control Wald test p-value: samplotype MOV10_overexpression vs control DataFrame with 6 rows and 6 columns baseMean log2FoldChange lfcSE stat pvalue padj
```

Check Individual gene plot to asses your analysis!!

```
# In essence we have the same data but now is a DeSeq2 data object
colData(dds_raw)
```

```
## DataFrame with 8 rows and 9 columns
##           SampleName      cell      dex     albut      Run avgLength
##           <factor> <factor> <factor> <factor> <factor> <integer>
## SRR1039508  GSM1275862  N61311    untrt    untrt SRR1039508     126
```

```

## SRR1039509 GSM1275863 N61311    trt      untrt SRR1039509      126
## SRR1039512 GSM1275866 N052611   untrt      untrt SRR1039512      126
## SRR1039513 GSM1275867 N052611   trt      untrt SRR1039513       87
## SRR1039516 GSM1275870 N080611   untrt      untrt SRR1039516     120
## SRR1039517 GSM1275871 N080611   trt      untrt SRR1039517     126
## SRR1039520 GSM1275874 N061011   untrt      untrt SRR1039520     101
## SRR1039521 GSM1275875 N061011   trt      untrt SRR1039521      98
##           Experiment     Sample     BioSample
##                 <factor> <factor> <factor>
## SRR1039508 SRX384345 SRS508568 SAMN02422669
## SRR1039509 SRX384346 SRS508567 SAMN02422675
## SRR1039512 SRX384349 SRS508571 SAMN02422678
## SRR1039513 SRX384350 SRS508572 SAMN02422670
## SRR1039516 SRX384353 SRS508575 SAMN02422682
## SRR1039517 SRX384354 SRS508576 SAMN02422673
## SRR1039520 SRX384357 SRS508579 SAMN02422683
## SRR1039521 SRX384358 SRS508580 SAMN02422677

head(assay(dds_raw))

##          SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG000000000003      679      448      873      408     1138
## ENSG000000000005       0         0         0         0         0
## ENSG00000000419      467      515      621      365      587
## ENSG00000000457      260      211      263      164      245
## ENSG00000000460       60        55        40        35        78
## ENSG00000000938       0         0         2         0         1
##          SRR1039517 SRR1039520 SRR1039521
## ENSG000000000003     1047      770      572
## ENSG000000000005       0         0         0
## ENSG00000000419      799      417      508
## ENSG00000000457      331      233      229
## ENSG00000000460       63        76        60
## ENSG00000000938       0         0         0

rowRanges(dds_raw)

## GRangesList object of length 63677:
## $ENSEG000000000003
## GRanges object with 17 ranges and 2 metadata columns:
##   seqnames      ranges strand | exon_id      exon_name
##     <Rle>      <IRanges> <Rle> | <integer> <character>
## [1]      X 99883667-99884983   - | 667145 ENSE00001459322
## [2]      X 99885756-99885863   - | 667146 ENSE00000868868
## [3]      X 99887482-99887565   - | 667147 ENSE00000401072
## [4]      X 99887538-99887565   - | 667148 ENSE00001849132
## [5]      X 99888402-99888536   - | 667149 ENSE00003554016
## ...
## [13]     ...     ...     ... | ...
## [14]     X 99890555-99890743   - | 667156 ENSE00003512331
## [15]     X 99891188-99891686   - | 667158 ENSE00001886883
## [16]     X 99891605-99891803   - | 667159 ENSE00001855382
## [17]     X 99891790-99892101   - | 667160 ENSE00001863395
## -----
## seqinfo: 722 sequences (1 circular) from an unspecified genome

```

```

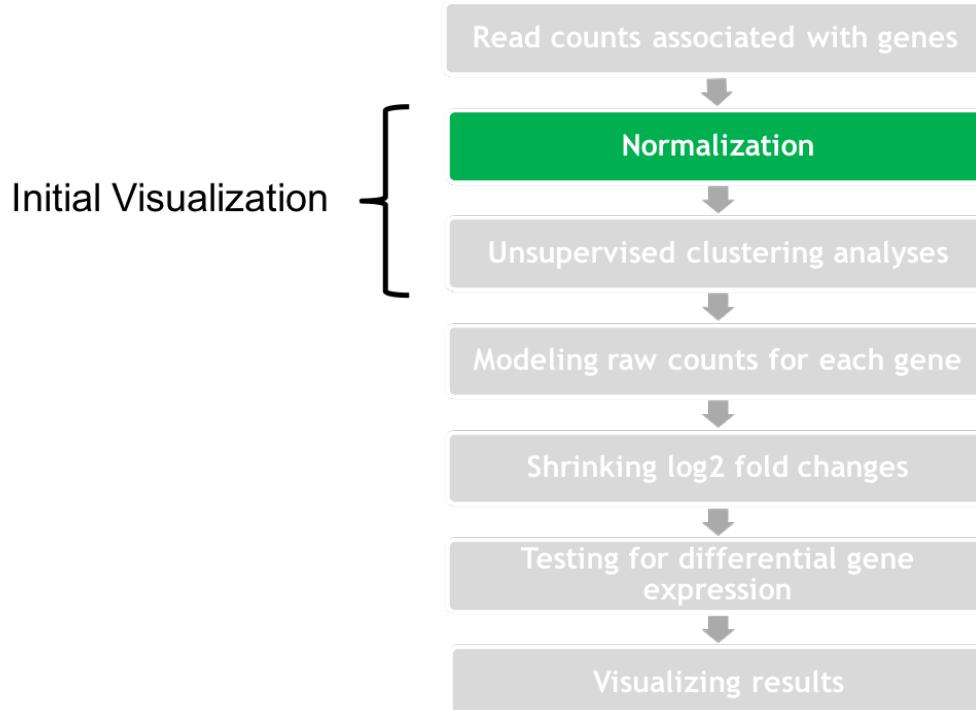
## 
## ...
## <63676 more elements>
# Check object attributes and class
#attributes(dds)
class(dds_raw)

## [1] "DESeqDataSet"
## attr(,"package")
## [1] "DESeq2"

```

1. Visual exploration of the data

There are two main paths on the RNA-seq differential expression analysis, first data exploration and second the proper DGE analysis. The **visualization** will imply to normalise the data in different ways and plot results to study the data distribution, correlation and clustering.



Gene count filtering

On our count table we have many genes with zero count. To reduce object size and computational time we can remove lowly expressed genes.

```

# Check counts on dds
head(counts(dds_raw))

##          SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG000000000003     679      448      873      408      1138
## ENSG000000000005      0         0         0         0         0
## ENSG00000000419     467      515      621      365      587
## ENSG00000000457     260      211      263      164      245
## ENSG00000000460      60       55       40       35       78

```

```

## ENSG00000000938      0      0      2      0      1
##                      SRR1039517 SRR1039520 SRR1039521
## ENSG000000000003    1047    770    572
## ENSG000000000005    0       0       0
## ENSG000000000419    799     417    508
## ENSG000000000457    331     233    229
## ENSG000000000460    63      76     60
## ENSG000000000938    0       0       0

# Genes removal with less than 10 reads on all the conditions. This will help DeSeq2 to perform faster.
dds_raw <- dds_raw[ rowSums(counts(dds_raw)) > 10, ]

# Check counts on dds after filtering
head(counts(dds_raw))

##          SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG000000000003    679     448    873    408    1138
## ENSG000000000419    467     515    621    365    587
## ENSG000000000457    260     211    263    164    245
## ENSG000000000460    60      55     40     35     78
## ENSG000000000971   3251    3679   6177   4252   6721
## ENSG000000001036   1433    1062   1733   881    1424
##          SRR1039517 SRR1039520 SRR1039521
## ENSG000000000003    1047    770    572
## ENSG000000000419    799     417    508
## ENSG000000000457    331     233    229
## ENSG000000000460    63      76     60
## ENSG000000000971   11027   5176   7995
## ENSG000000001036   1439    1359   1109

```

Normalization

```

# Investigate different sequencing library sizes. DGE software will use this sequencing depth information
colSums(counts(dds_raw))

```

Sequencing depth normalization

```

## SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517 SRR1039520
## 20633673   18805036   25342811   15160577   24442836   30811815   19121708
## SRR1039521
## 21160391

```

Calculate the size factor and add it to the data set.

```

dds_raw <- estimateSizeFactors(dds_raw)
sizeFactors(dds_raw)

```

```

## SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517 SRR1039520
## 1.0236767 0.8962366 1.1795988 0.6699588 1.1776734 1.3991183 0.9207797
## SRR1039521
## 0.9446040

```

`counts()` allows to immediately retrieve the **normalized** read counts.

```

counts_normalized <- counts(dds_raw, normalized = TRUE)

```

Now take a look at the sum of the total depth after normalization

```

colSums(counts(dds_raw, normalized=TRUE))

```

```

## SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517 SRR1039520

```

```

##    20156435    20982222    21484263    22629118    20755190    22022309    20766865
## SRR1039521
##    22401336

# Observe now the normalized counts
head(counts_normalized)

##          SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG000000000003 663.29536 499.86798 740.08214 608.99265 966.31203
## ENSG000000000419 456.19872 574.62502 526.45018 544.80960 498.44039
## ENSG000000000457 253.98644 235.42889 222.95716 244.79116 208.03730
## ENSG000000000460 58.61226 61.36772 33.90983 52.24202 66.23228
## ENSG000000000971 3175.80737 4104.94265 5236.52621 6346.65873 5707.01505
## ENSG000000001036 1399.85603 1184.95491 1469.14358 1315.00619 1209.16373
##          SRR1039517 SRR1039520 SRR1039521
## ENSG000000000003 748.32844 836.24781 605.54477
## ENSG000000000419 571.07395 452.87706 537.79151
## ENSG000000000457 236.57757 253.04641 242.42964
## ENSG000000000460 45.02836 82.53874 63.51868
## ENSG000000000971 7881.39228 5621.32292 8463.86436
## ENSG000000001036 1028.50490 1475.92308 1174.03697

```

Log² transformation of read counts. Log² transformation of the data make normalization easily interpretable and comparable. Simply using the log2 function, after adding 1 pseudocount, to avoid taking the log of zero.

```

# Log^2 normalization adding 1 pseudocount.
counts_log_normalized <- log2(counts_normalized + 1)

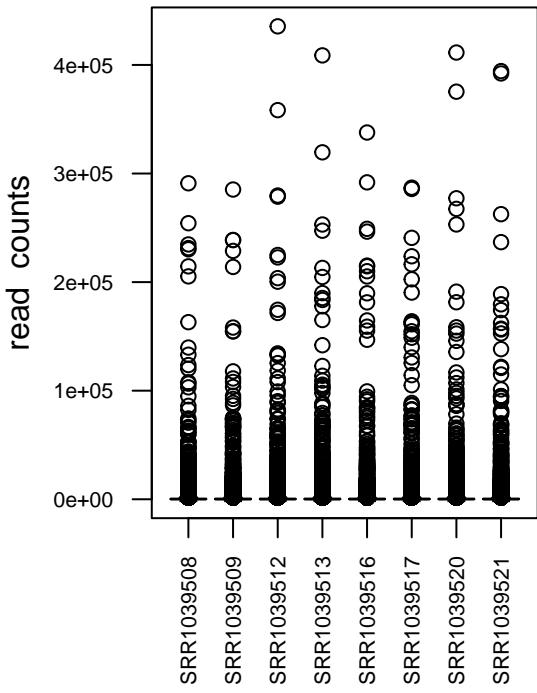
par(mfrow=c(1,2)) # to plot the following two images side by side each other

# first , boxplots of non -transformed read counts (one per sample)
boxplot(counts_normalized , notch = TRUE , las=2, cex.axis = 0.7,
        main = "untransformed read counts", ylab = "read counts")

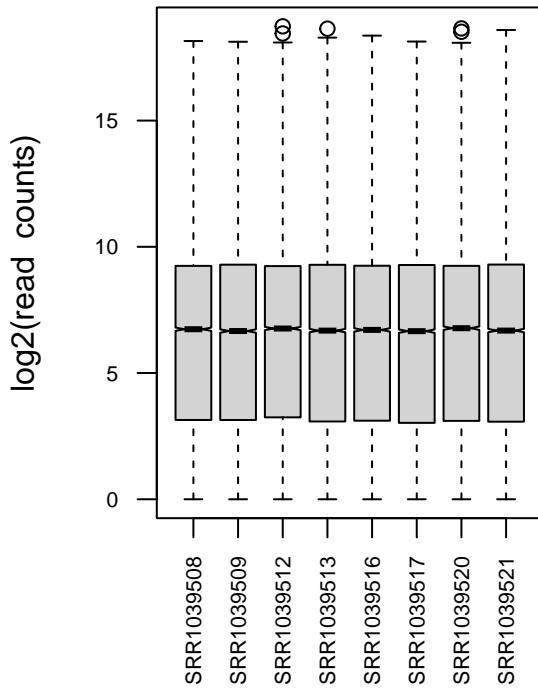
# box plots of log^2 -transformed read counts
boxplot(counts_log_normalized , notch = TRUE , las=2, cex.axis = 0.7,
        main = "log2 -transformed read counts",
        ylab = "log2(read counts)")

```

untransformed read counts



log2 -transformed read counts



Data transformation When the expected amount of variance is approximately the same across different mean values, the data is said to be homoskedastic. For RNA-seq counts, however, the expected variance grows with the mean.

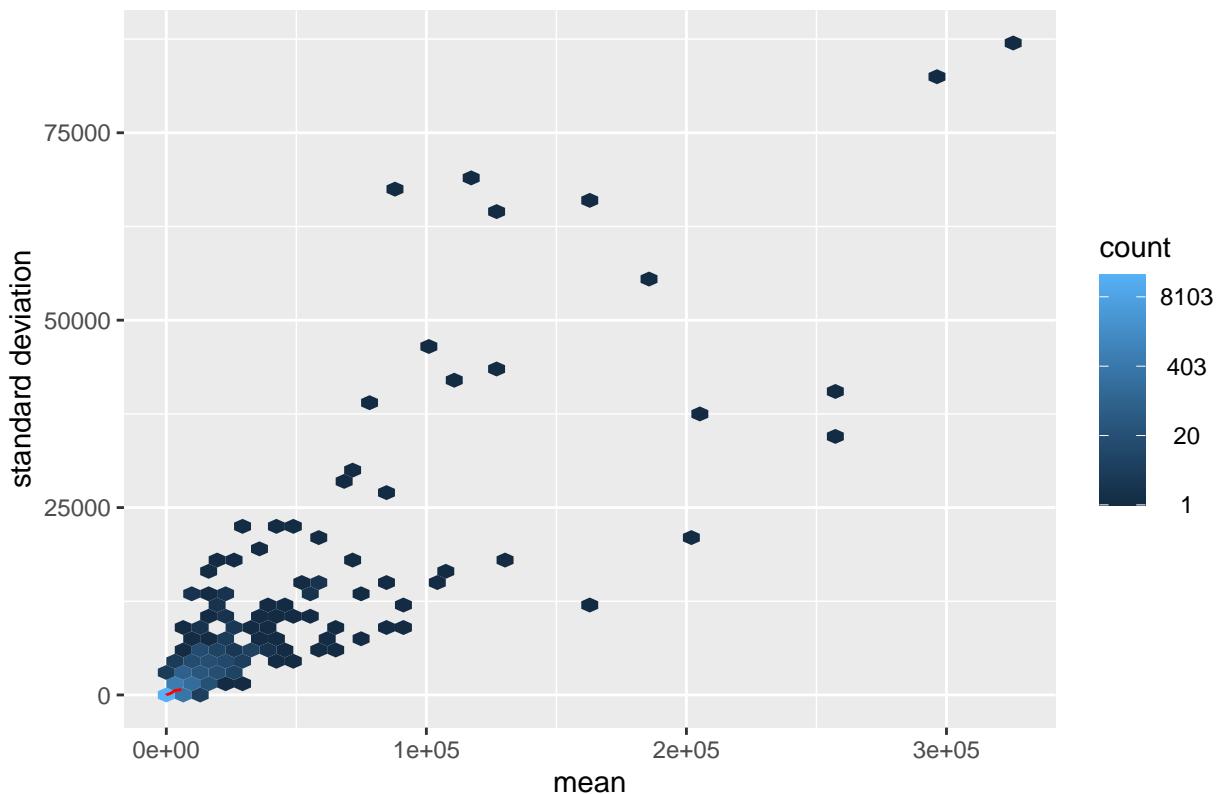
```
# If needed, install the vsn package
# BiocManager::install("vsn")
# biocLite("vsn")
# if (!require(vsn)) {
#   install.packages("vsn")
#   require(vsn)
# }
library("vsn")

# If needed, install the ggplot2 package
# BiocManager::install("ggplot2", force = TRUE)
# if (!require(ggplot2)) {
#   install.packages("ggplot2")
#   require(ggplot2)
# }
library(ggplot2)

# BiocManager::install("hexbin")
library("hexbin")

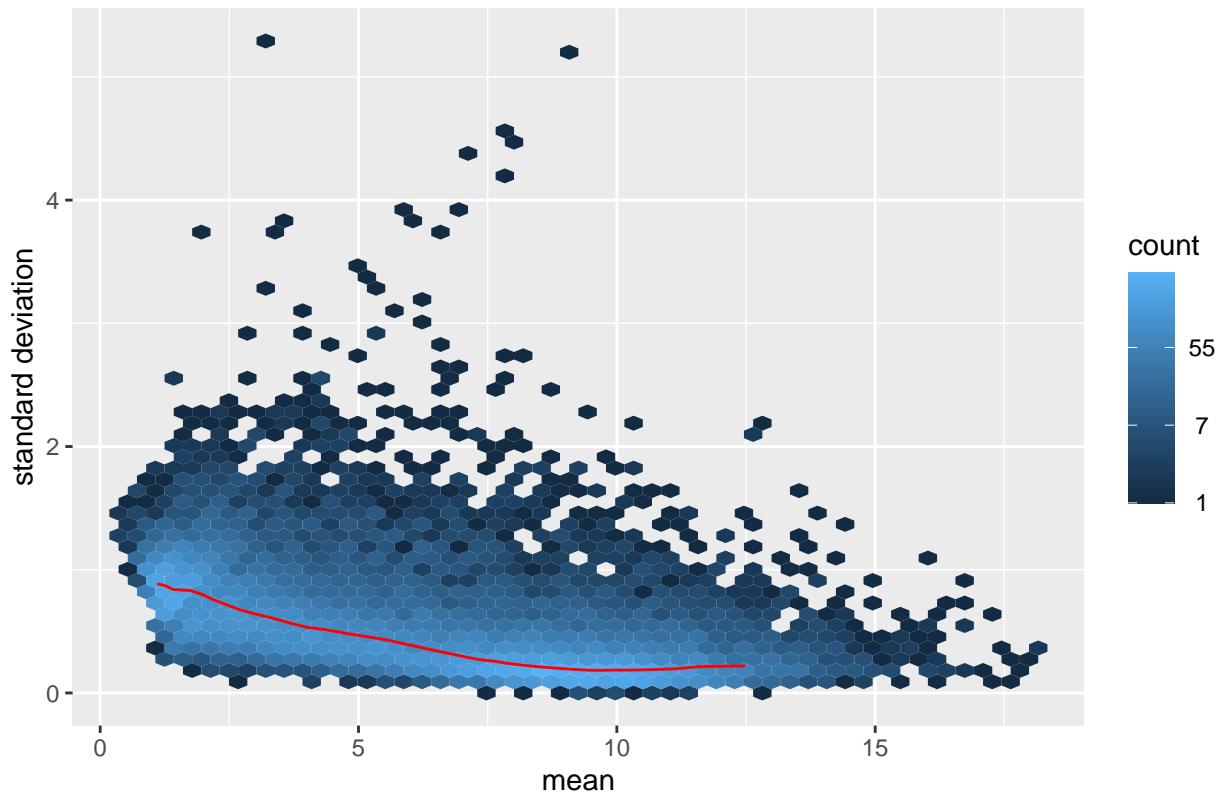
# When the expected amount of variance is approximately the same across different mean values, the data
SdPlot <- meanSdPlot(counts_normalized, ranks = FALSE, plot = FALSE)
SdPlot$gg + ggtitle("sequencing depth normalized") + ylab("standard deviation")
```

sequencing depth normalized



```
# The logarithm with a small pseudocount amplifies differences when the values are close to 0. The low
SdPlot_log <- meanSdPlot(counts_log_normalized, ranks = FALSE, plot = FALSE)
SdPlot_log$gg + ggtitle("sequencing depth normalized log2(read counts)") + ylab("standard deviation")
```

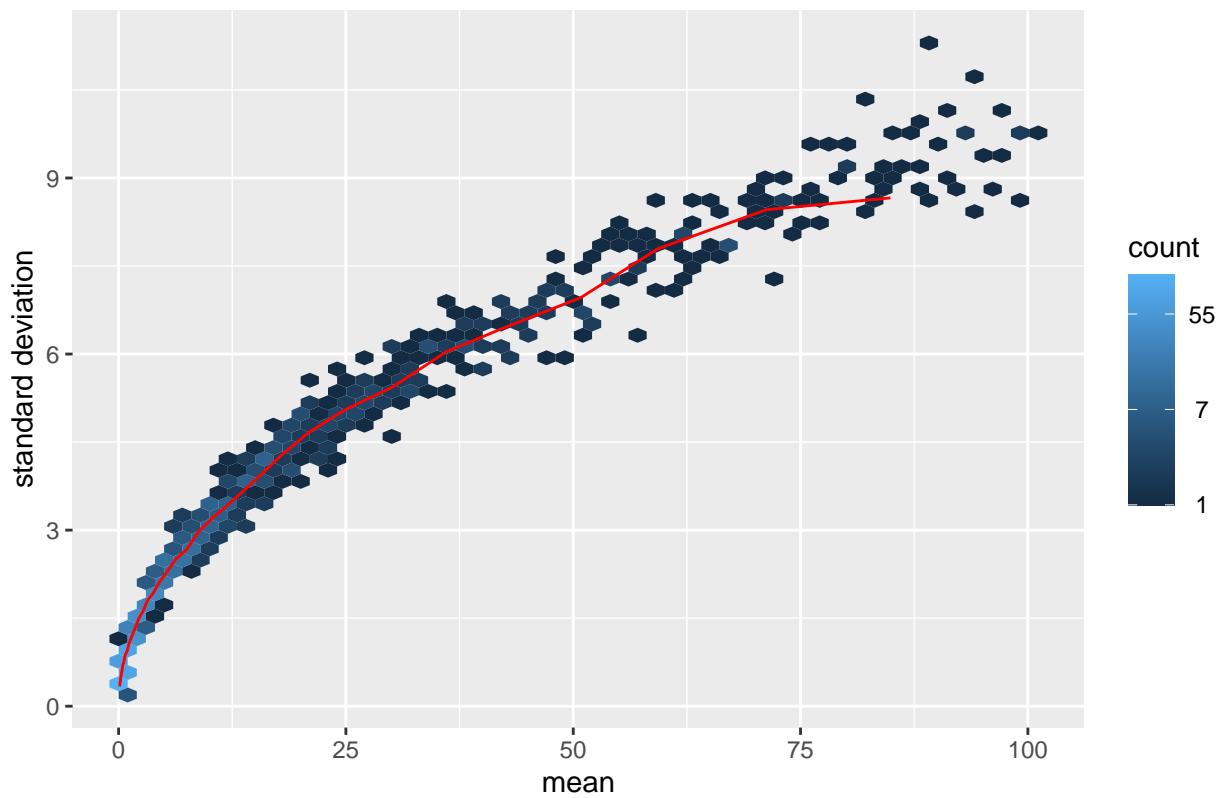
sequencing depth normalized log2(read counts)



```
# We can see this property of count with simulated data. Poisson counts in a range of lambda 0.1 to 100
lambda <- 10^seq(from = -1, to = 2, length = 1000)
cts <- matrix(rpois(1000*100, lambda), ncol = 100)

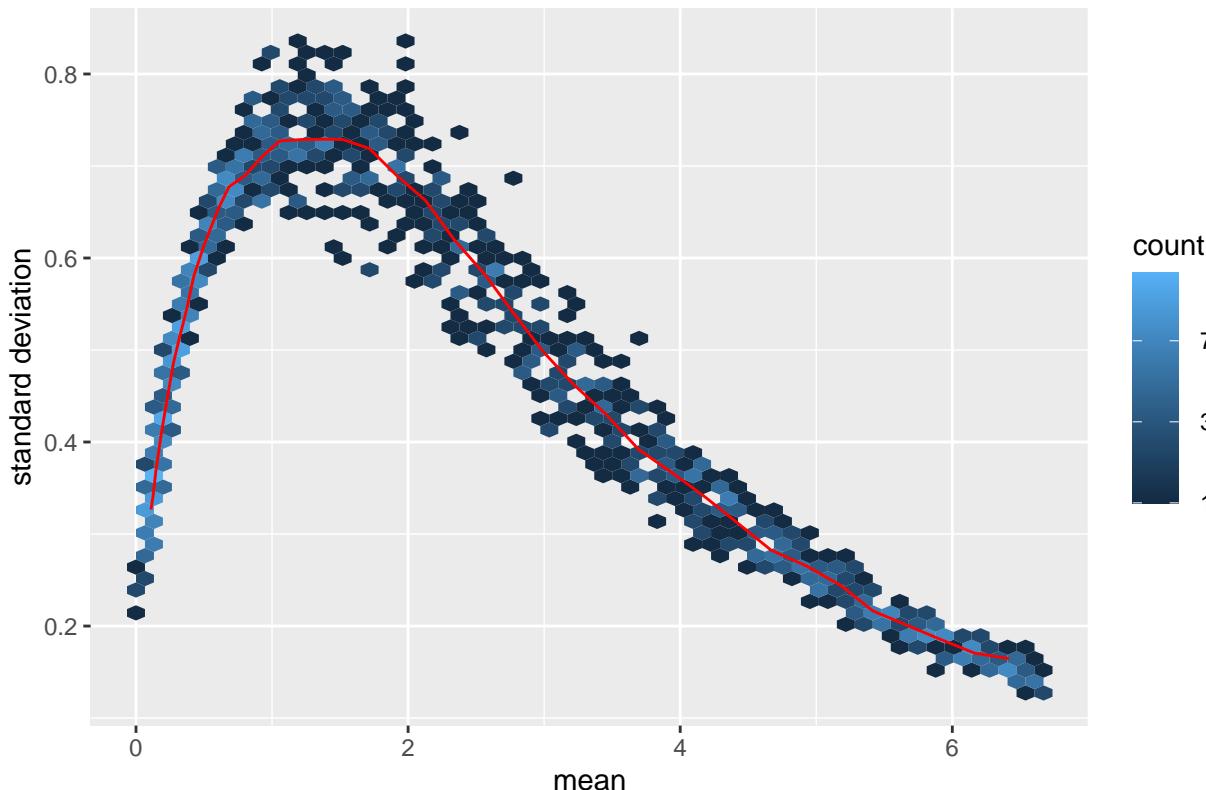
# Standard deviation of each row against the mean on raw counts
sim_SdPlot <- meanSdPlot(cts, ranks = FALSE, plot = FALSE)
sim_SdPlot$gg + ggtitle("simulated data sequencing depth normalized") + ylab("standard deviation")
```

simulated data sequencing depth normalized)



```
# Standard deviation of each row against the mean on log2 pseudocount transformed data
log.cts.one <- log2(cts + 1)
sim_SdPlot_log <- meanSdPlot(log.cts.one, ranks = FALSE, plot = FALSE)
sim_SdPlot_log$gg + ggtitle("simulated data sequencing depth normalized log2(read counts)") + ylab("standard deviation")
```

simulated data sequencing depth normalized log2(read counts)



DeSeq2 data transformation As a solution, DESeq2 offers two transformations for count data that stabilize the variance across the mean: the variance stabilizing transformation (VST) for negative binomial data with a dispersion-mean trend (Anders and Huber 2010), implemented in the `vst` function, and the regularized-logarithm transformation or rlog (Love, Huber, and Anders 2014).

For genes with high counts, both the VST and the rlog will give similar result to the ordinary \log_2 transformation of normalized counts. For genes with lower counts, however, the values are shrunk towards a middle value. The VST or rlog-transformed data then become approximately homoskedastic (more flat trend in the `meanSdPlot`), and can be used directly for computing distances between samples, making PCA plots, or as input to downstream methods which perform best with homoskedastic data.

Which transformation to choose? The VST is much faster to compute and is less sensitive to high count outliers than the rlog. The rlog tends to work well on small datasets ($n < 30$), potentially outperforming the VST when there is a wide range of sequencing depth across samples (an order of magnitude difference). We therefore recommend the VST for medium-to-large datasets ($n > 30$). You can perform both transformations and compare the `meanSdPlot` or PCA plots generated, as described below.

Note that the two transformations offered by DESeq2 are provided for applications other than differential testing. For differential testing we recommend the `DESeq` function applied to **raw counts**, as described later in this workflow, which also takes into account the dependence of the variance of counts on the mean value during the dispersion estimation step.

- **Variance stabilizing transformation**

```
vsd <- vst(dds_raw, blind = FALSE)
head(assay(vsd), 3)

##           SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG000000000003    9.819191   9.523047   9.939055   9.727694  10.242102
```

```

## ENSG00000000419    9.431670    9.666481    9.575795    9.611069    9.520155
## ENSG00000000457    8.899577    8.837557    8.794049    8.869235    8.739868
##          SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003    9.951334    10.075948   9.721674
## ENSG000000000419    9.659998    9.424459   9.597692
## ENSG000000000457    8.841488    8.896509   8.861321

```

- The rlog

```

rld <- rlog(dds_raw, blind = FALSE)
head(assay(rld), 3)

```

```

##          SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003    9.385710    9.052987    9.516677    9.285691    9.838666
## ENSG000000000419    8.869621    9.138138    9.035993    9.075431    8.972129
## ENSG000000000457    7.961549    7.881275    7.823774    7.921721    7.751268
##          SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003    9.530143    9.663011    9.277774
## ENSG000000000419    9.131721    8.861576    9.060774
## ENSG000000000457    7.886366    7.957315    7.912085

```

Sample correlation using all the different data transformations We will study the correlation on the first two samples using different data transformations.

```

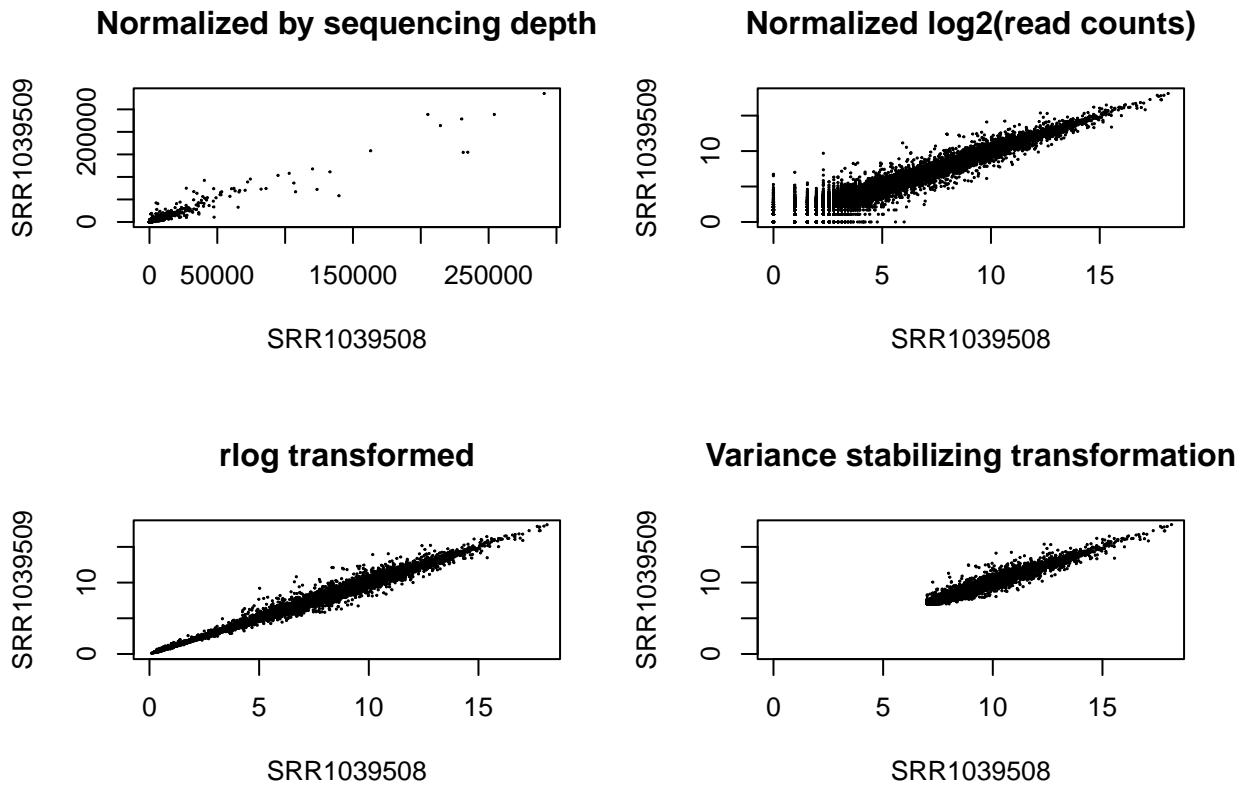
par(mfrow=c(2,2))
# Raw data normalized by sequencing depth
plot(counts_normalized[,1:2], cex=.1, main = "Normalized by sequencing depth")

# Log^2 normalization adding 1 pseudocount.
plot(counts_log_normalized[,1:2], cex=.1, main = "Normalized log2(read counts)")

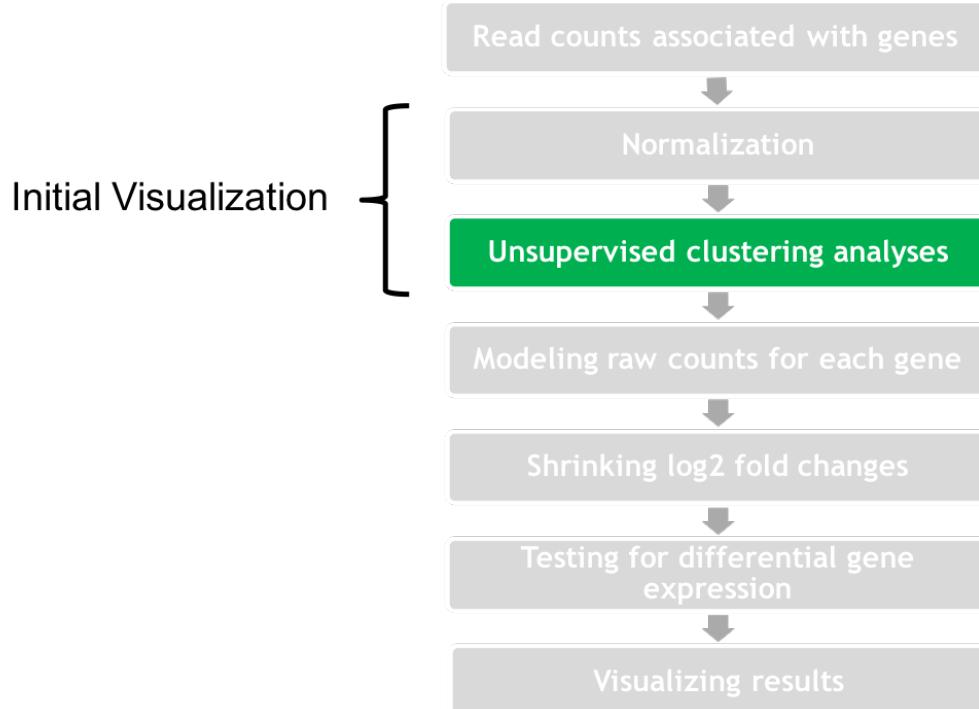
# rlog transformed
rlog_norm_counts <- assay(rld)
plot(rlog_norm_counts[,1:2], cex=.1, main = "rlog transformed", xlim=c(0,18), ylim=c(0,18))

# Variance stabilizing transformation
vsd_norm_counts <- assay(vsd)
plot(vsd_norm_counts[,1:2], cex=.1, main = "Variance stabilizing transformation", xlim=c(0,18), ylim=c(0,18))

```



PCA & hierarchical clustering Identifying outliers samples and sources of variation in the data.



- Variance stabilizing transformation

```
# If needed, install the pheatmap and RColorBrewer packages
# BiocManager::install("pheatmap")
```

```

# if (!require(pheatmap)) {
#   install.packages("pheatmap")
#   require(pheatmap)
# }
library(pheatmap)

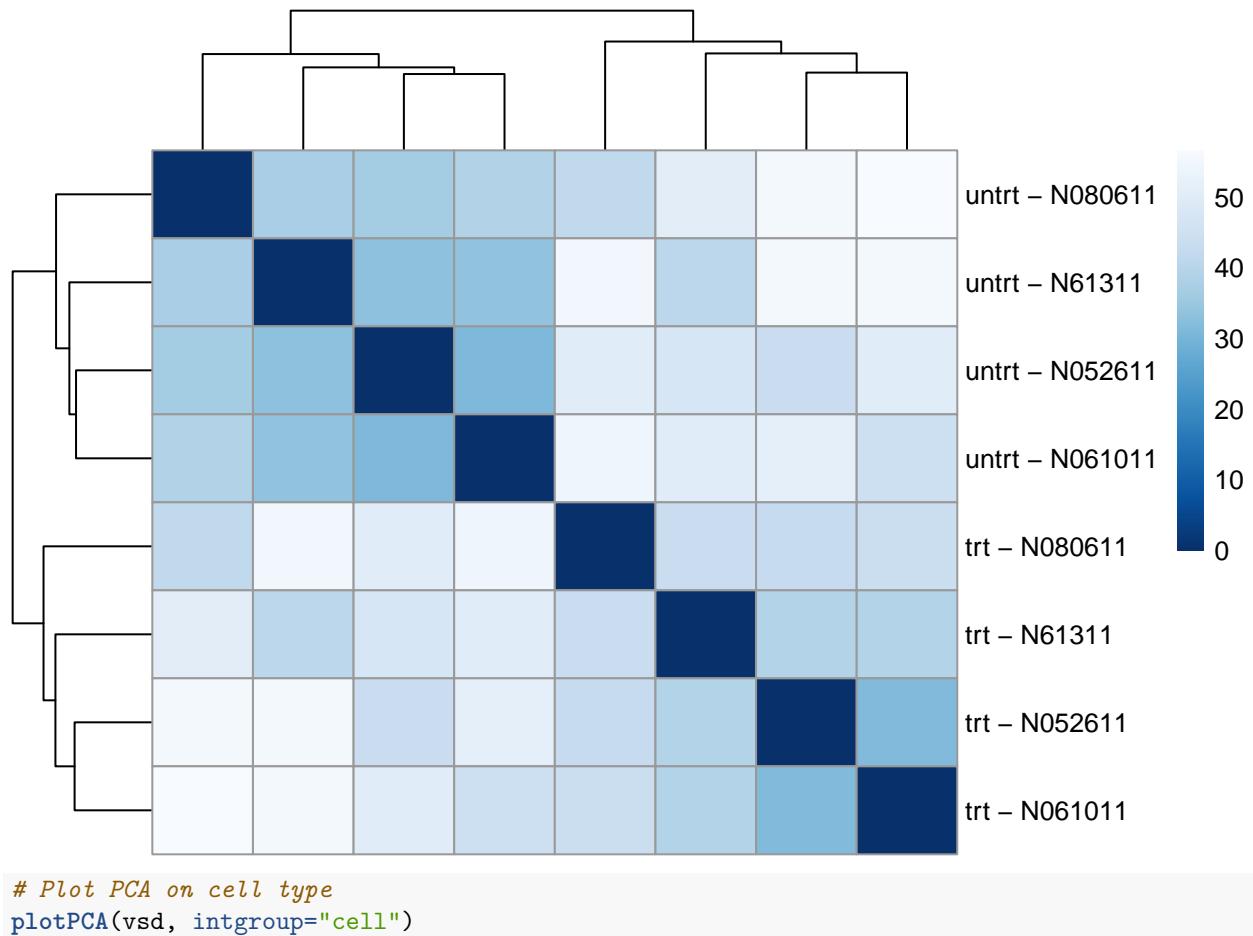
# BiocManager::install("RColorBrewer")
# if (!require(RColorBrewer)) {
#   install.packages("RColorBrewer")
#   require(RColorBrewer)
# }
library(RColorBrewer)

# Sample distances with VST
sampleDists_vsd <- dist(t(assay(vsd)))
sampleDists_vsd

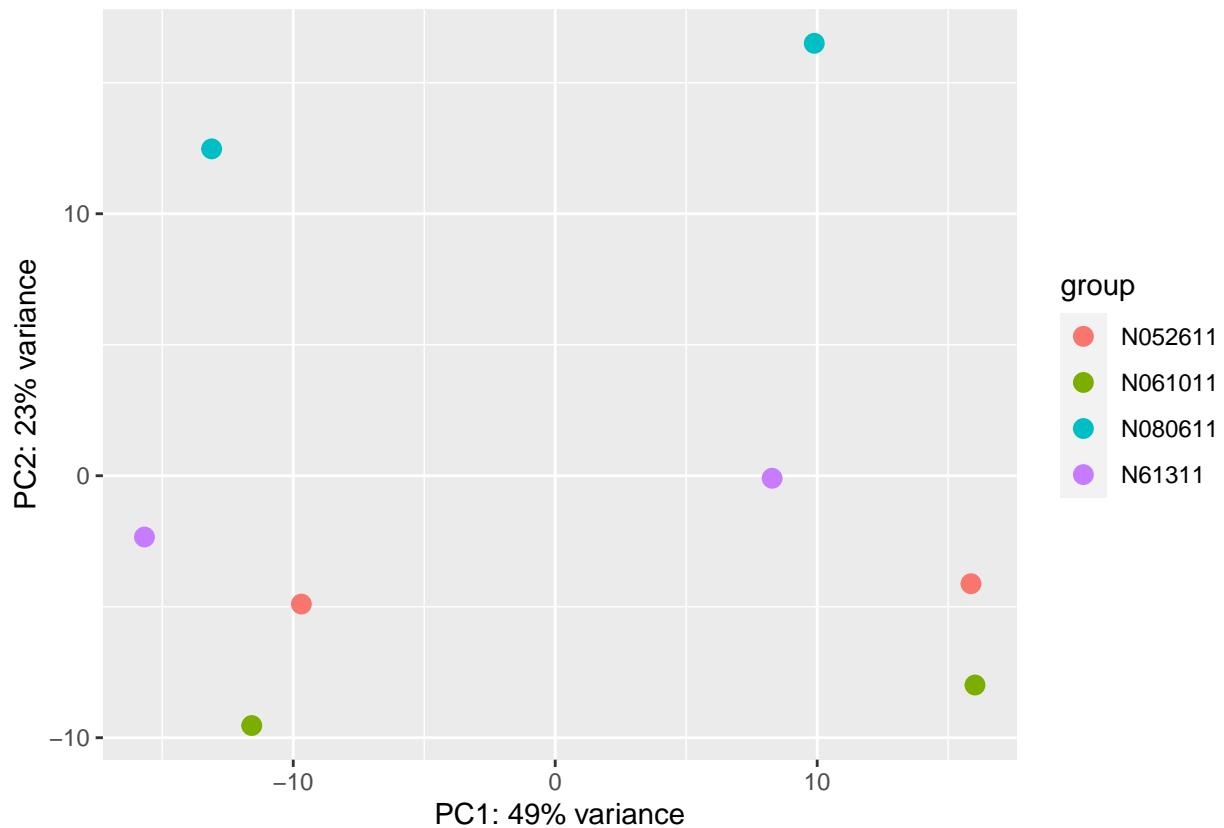
##          SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517
## SRR1039509    40.71891
## SRR1039512    33.32015    47.68646
## SRR1039513    55.32751    39.30305    43.74533
## SRR1039516    37.41412    50.83267    36.64253    55.69234
## SRR1039517    55.19399    43.71746    50.39249    42.60601    41.60202
## SRR1039520    33.64095    50.03632    30.93601    51.49151    39.04477    54.24523
## SRR1039521    55.43354    39.14690    50.37847    31.52966    56.64737    44.12058
##          SRR1039520
##          SRR1039509
##          SRR1039512
##          SRR1039513
##          SRR1039516
##          SRR1039517
##          SRR1039520
##          SRR1039521    44.86001

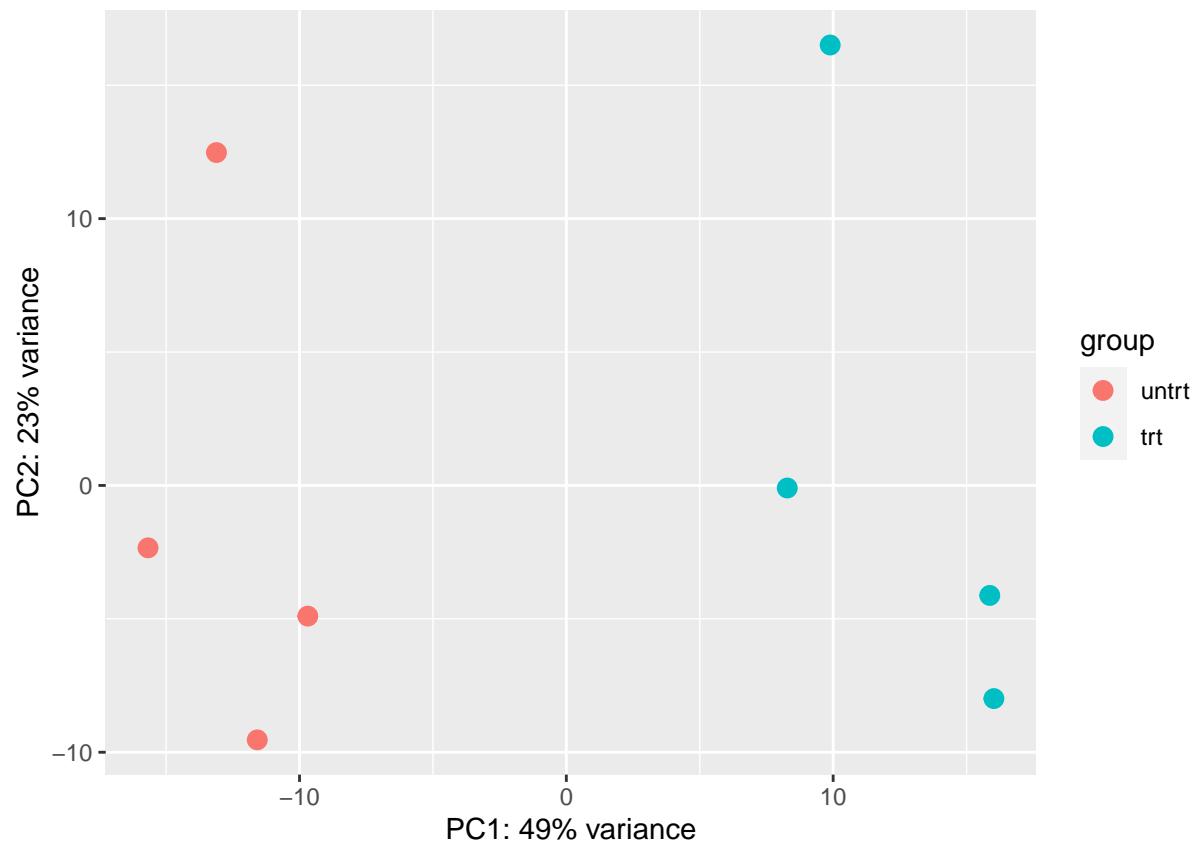
# Transform sample distances to matrix
sampleDistMatrix_vsd <- as.matrix( sampleDists_vsd )
# Modify row names joining treatment and cell type
rownames(sampleDistMatrix_vsd) <- paste( vsd$dex, vsd$cell, sep = " - " )
# Remove col names
colnames(sampleDistMatrix_vsd) <- NULL
# Colors palette
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
# Draw heatmap
heatmap <- pheatmap(sampleDistMatrix_vsd,
                      clustering_distance_rows = sampleDists_vsd,
                      clustering_distance_cols = sampleDists_vsd,
                      col = colors)

```

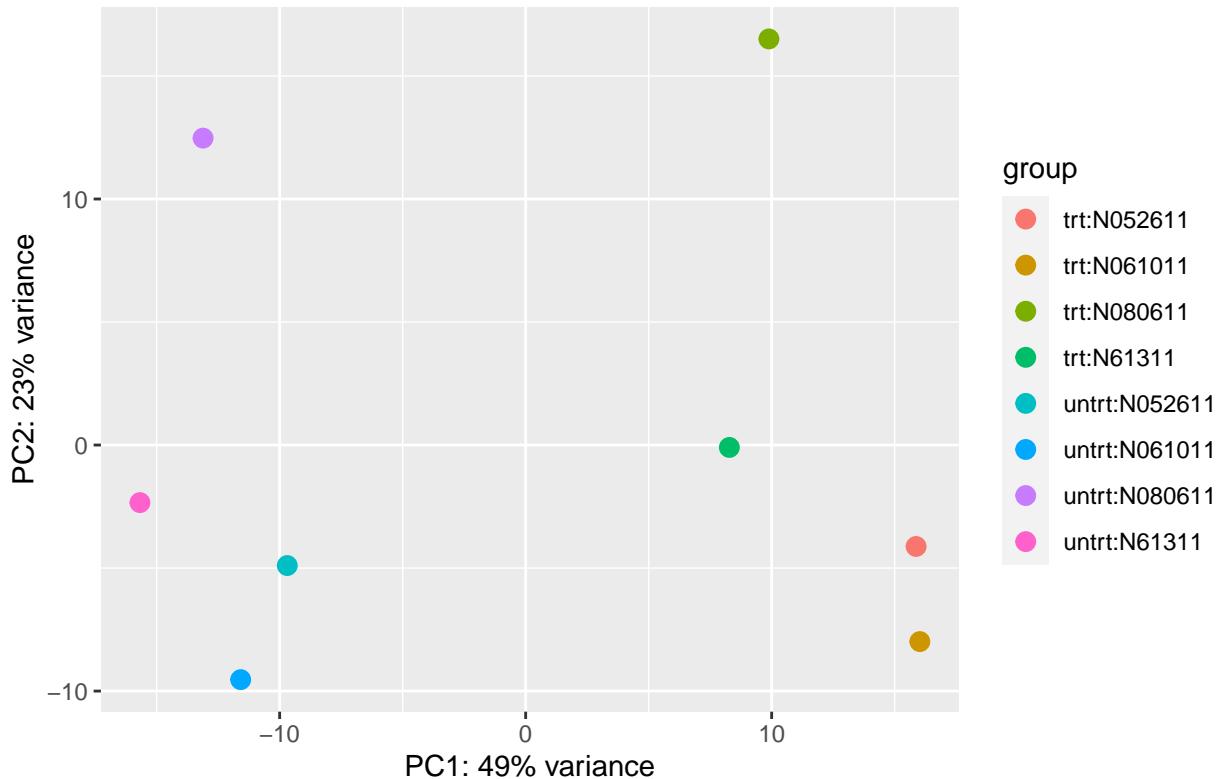


```
# Plot PCA on cell type
plotPCA(vsd, intgroup="cell")
```





```
# Plot PCA on cell type and dexamethasone treatment
plotPCA(vsd, intgroup=c("dex", "cell"))
```

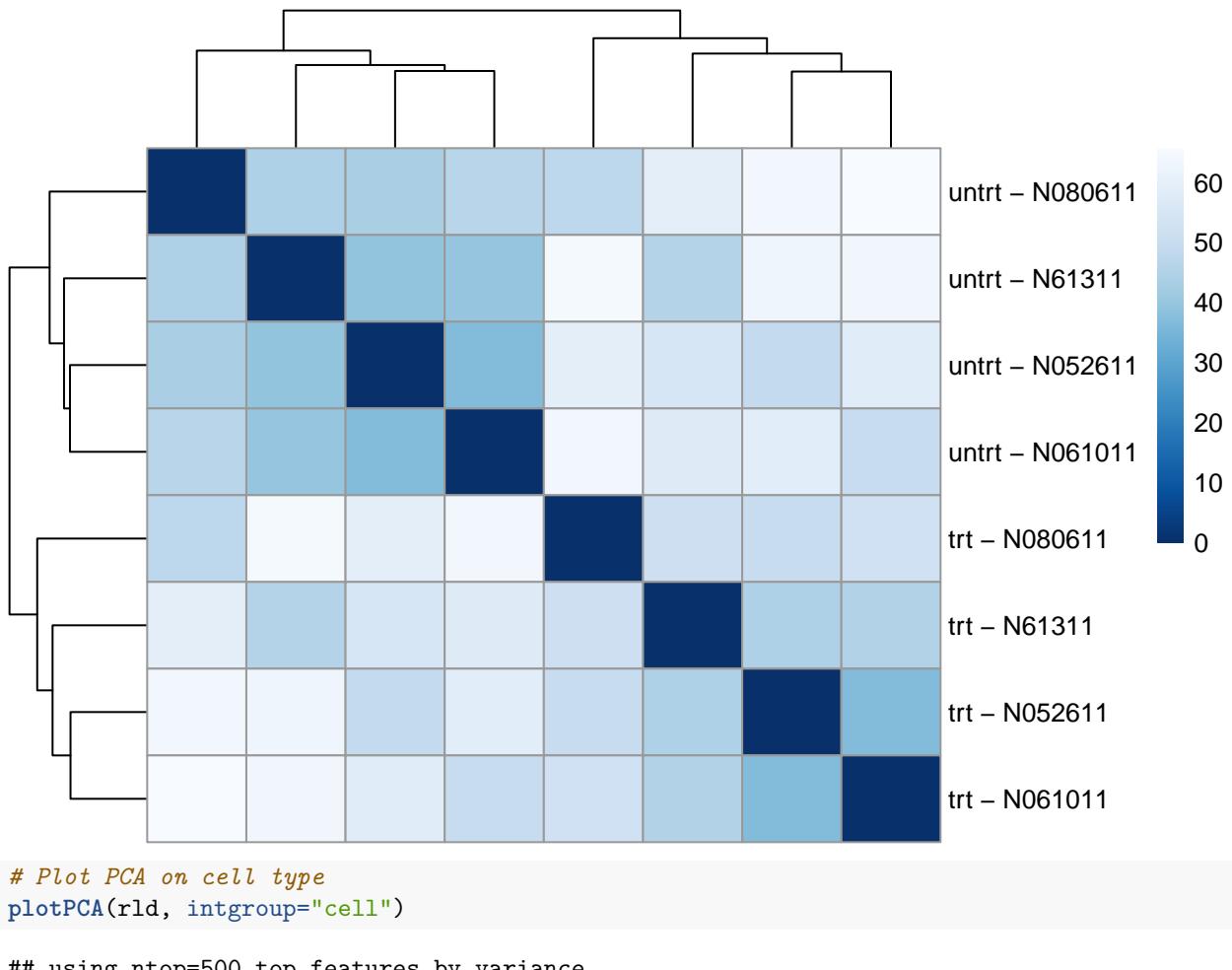


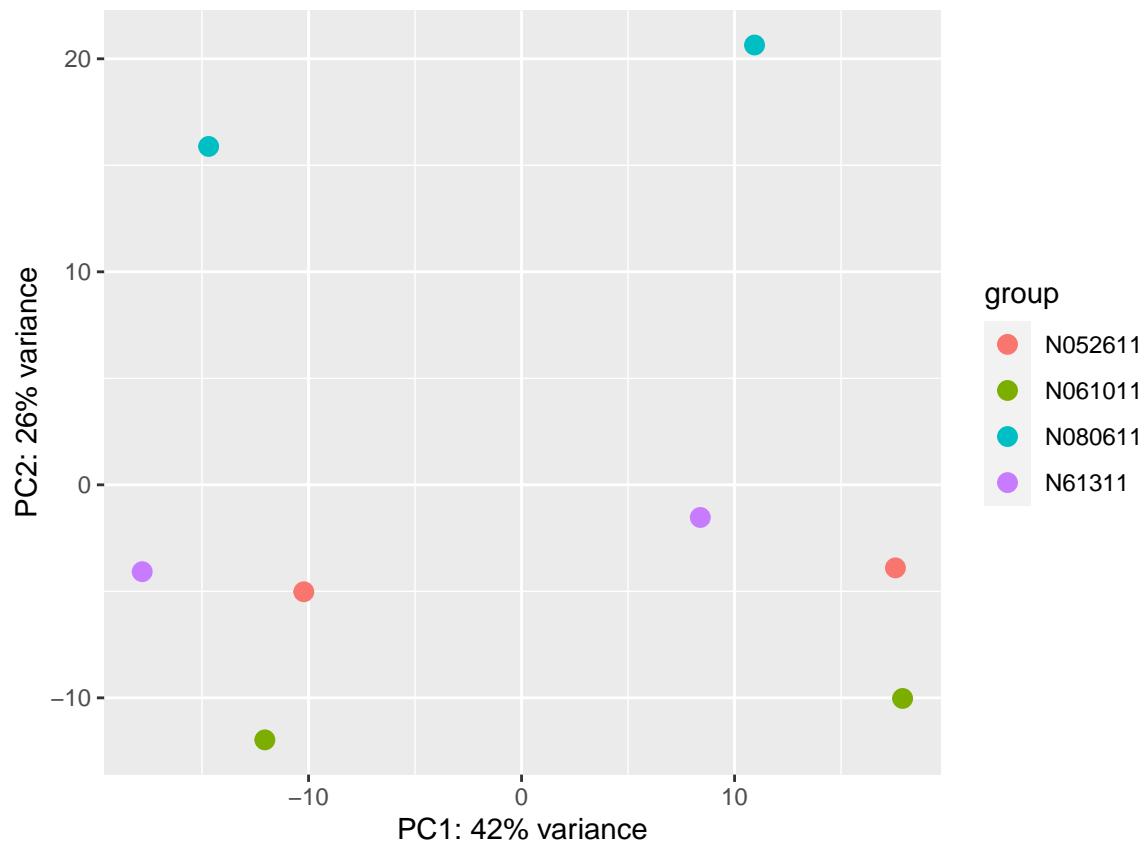
- The rlog

```
# Sample distances with rld
sampleDists_rld <- dist(t(assay(rld)))
sampleDists_rld

##          SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517
## SRR1039509    45.76449
## SRR1039512    39.39689    55.05195
## SRR1039513    62.75295    44.63162    48.80591
## SRR1039516    44.67357    59.22327    43.73393    63.88698
## SRR1039517    64.67402    51.61886    59.40815    50.04076    47.57564
## SRR1039520    39.68018    57.57696    36.83867    58.59540    46.55154    63.76998
## SRR1039521    63.48227    45.15104    57.99941    36.58571    65.69836    52.48221
##          SRR1039520
## SRR1039509
## SRR1039512
## SRR1039513
## SRR1039516
## SRR1039517
## SRR1039520
## SRR1039521    50.21932

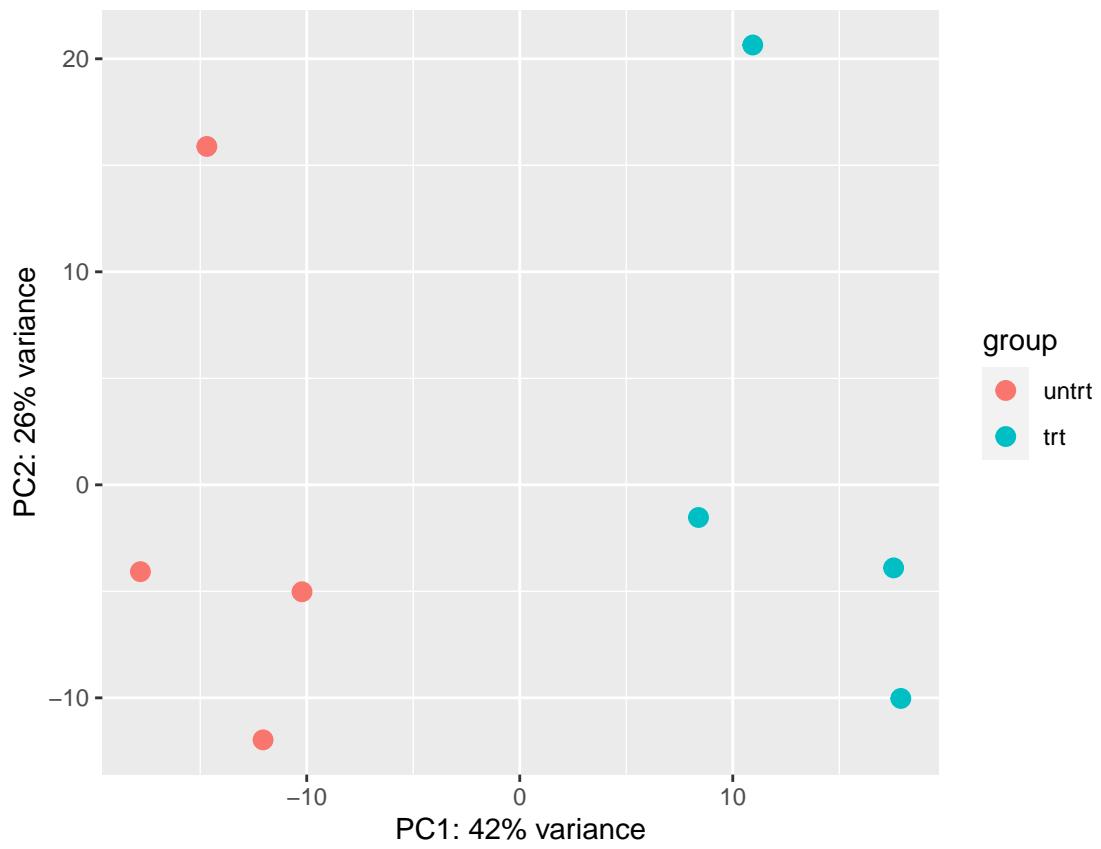
# Transform sample distances to matrix
sampleDistMatrix_rld <- as.matrix( sampleDists_rld )
# Modify row names joining treatment and cell type
rownames(sampleDistMatrix_rld) <- paste( rld$dex, rld$cell, sep = " - " )
# Remove col names
colnames(sampleDistMatrix_rld) <- NULL
# Colors palette
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
# Draw heatmap
heatmap <- pheatmap(sampleDistMatrix_rld,
                      clustering_distance_rows = sampleDists_rld,
                      clustering_distance_cols = sampleDists_rld,
                      col = colors)
```





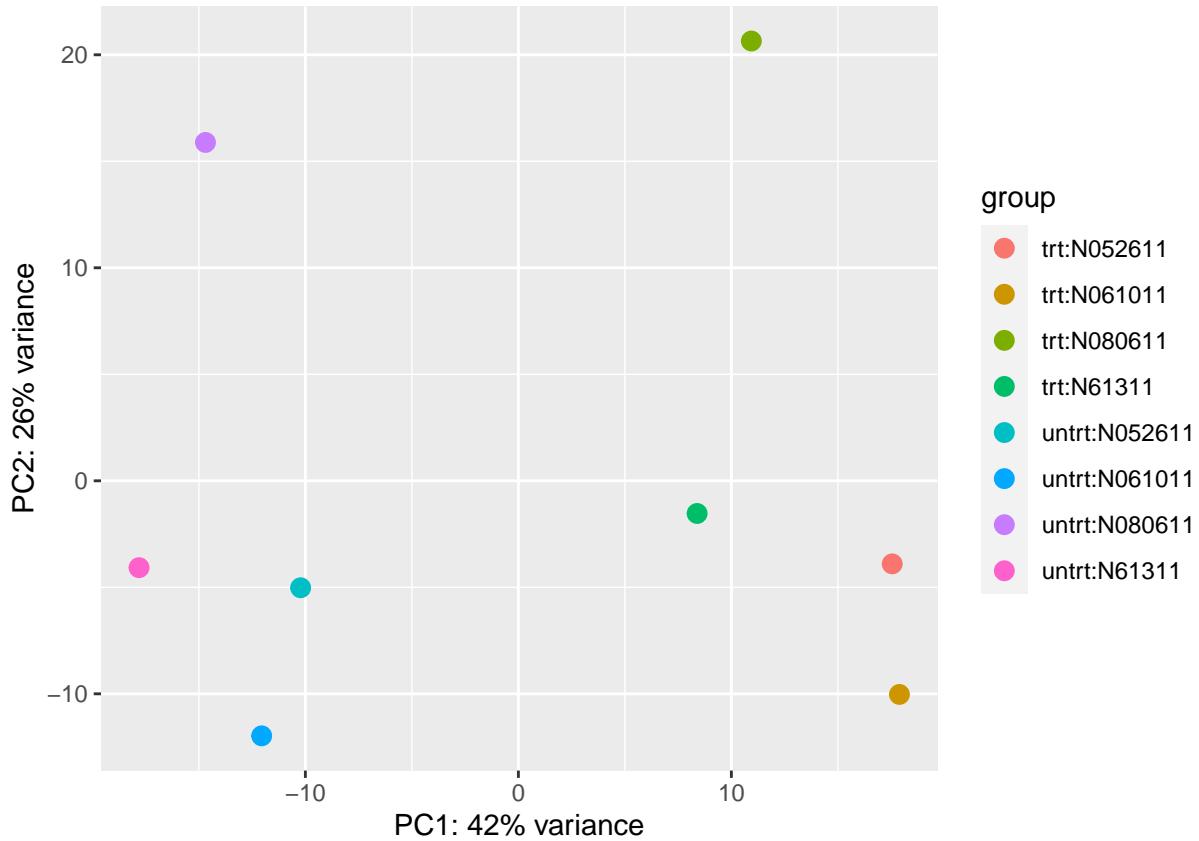
```
# Plot PCA on dexamethasone treatment
plotPCA(rld, intgroup="dex")
```

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



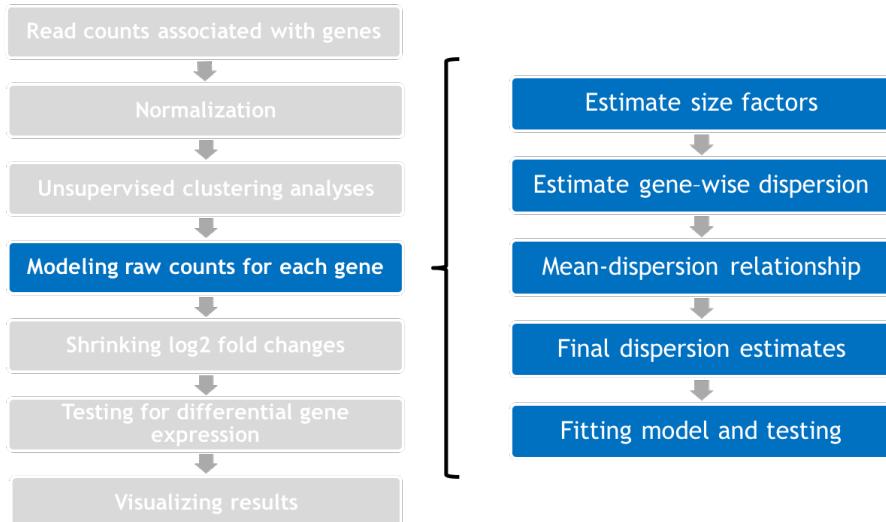
```
# Plot PCA on cell type and dexamethasone treatment
plotPCA(rld, intgroup=c("dex", "cell"))
```

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



2. Differential gene expression

Now after the data exploration we can run the differential expression pipeline on the **raw counts** with a single call to the function DESeq. The null hypothesis is that there is no systematic difference between the average read count values of the different conditions for a given gene. We will calculate the fold change of read counts, assuming de differences in sequencing depth and variability. For our data we will like to test the effect of dexamethasone treatment versus the untreated cells (untreated used as denominator for the fold change calculation).

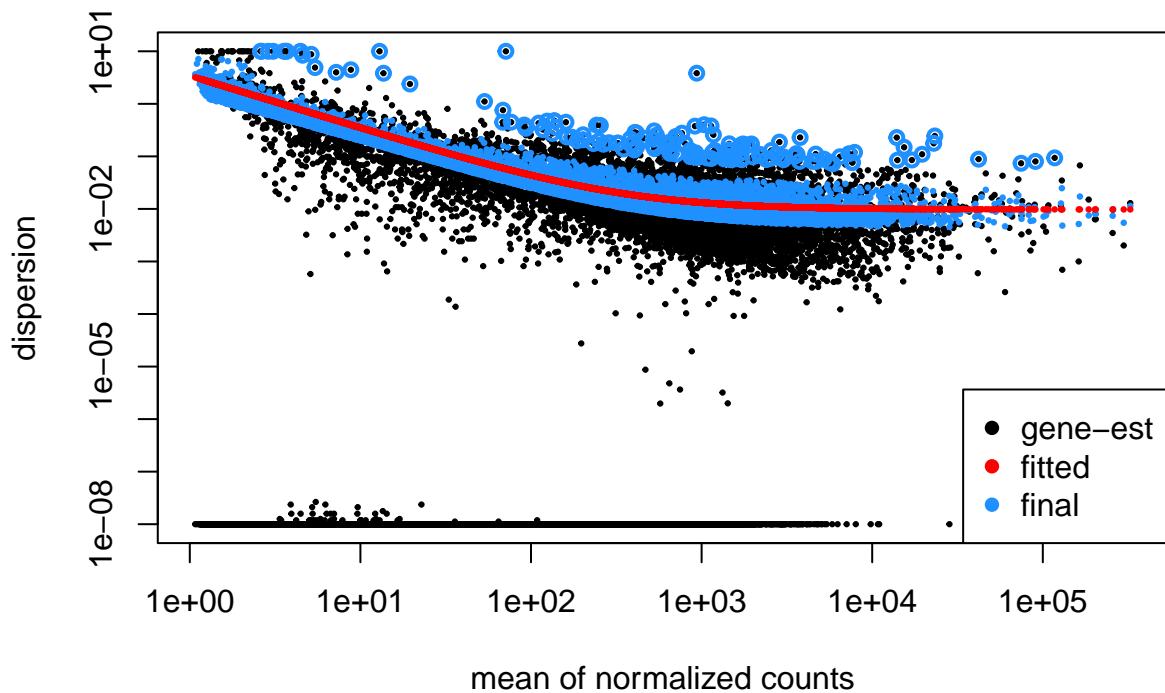


```

# Run DGE DeSeq2 pipeline
dds_DGE <- DESeq(dds_raw)

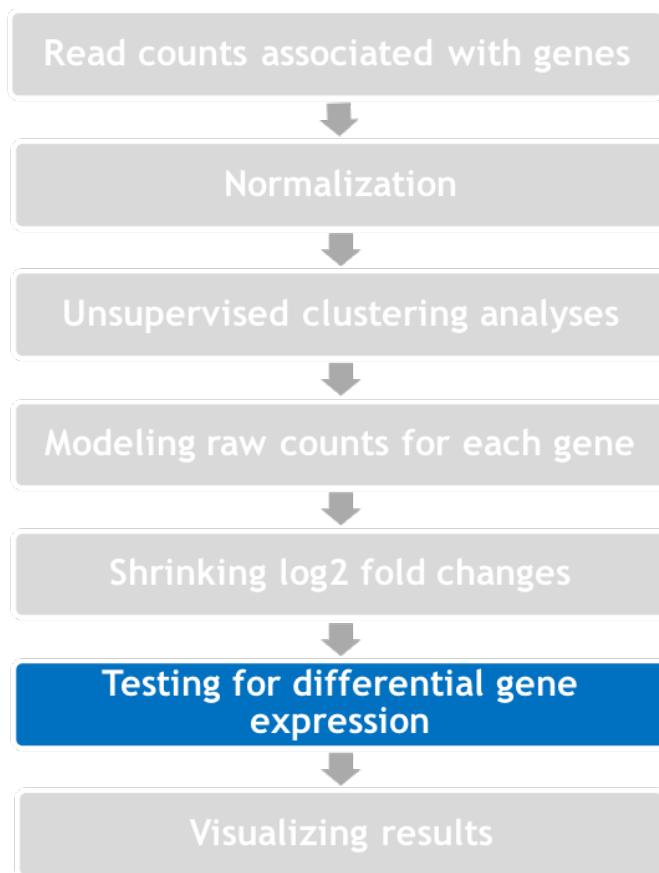
## using pre-existing size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
# Mean-dispersion relationship
par(mfrow=c(1,1))
plotDispEts(dds_DGE)

```



Building the results table

Now we can extract the results for the selected contrast, just calling `results()`.



```

# Calling results() will build the base means across samples, log2 fold variation, standard errors, p v
dds_DGE_results <- results(dds_DGE)
head(dds_DGE_results)

## log2 fold change (MLE): dex trt vs untrt
## Wald test p-value: dex trt vs untrt
## DataFrame with 6 rows and 6 columns
##           baseMean log2FoldChange      lfcSE      stat      pvalue
##           <numeric>     <numeric> <numeric> <numeric>   <numeric>
## ENSG00000000003  708.5839    -0.3812438 0.1006569 -3.787556 1.52136e-04
## ENSG00000000419  520.2833     0.2068250 0.1119673  1.847190 6.47195e-02
## ENSG00000000457  237.1568     0.0379405 0.1425055  0.266239 7.90056e-01
## ENSG00000000460  57.9312    -0.0887308 0.2839549 -0.312482 7.54674e-01
## ENSG00000000971  5817.1912    0.4264088 0.0889024  4.796371 1.61566e-06
## ENSG00000001036 1282.0737    -0.2410581 0.0890373 -2.707383 6.78160e-03
##           padj
##           <numeric>
## ENSG00000000003 1.25654e-03
## ENSG00000000419 1.91389e-01
## ENSG00000000457 9.07267e-01
## ENSG00000000460 8.89731e-01
## ENSG00000000971 2.07705e-05
## ENSG00000001036 3.30256e-02

# Result object can be filter like a data frame. We have 4023 significant genes with a p adjusted < 0.05
table(dds_DGE_results$padj < 0.05)

```

```

##  

## FALSE TRUE  

## 13718 4023  

# rownames(subset(dds_DGE_results, padj < 0.01))  

# We can also access to the metadata contained on the columns and check the pairwise contrast studied (.  

mcols(dds_DGE_results, use.names = TRUE)  

## DataFrame with 6 rows and 2 columns  

## type description  

## <character> <character>  

## baseMean intermediate mean of normalized c..  

## log2FoldChange results log2 fold change (ML..  

## lfcSE results standard error: dex ..  

## stat results Wald statistic: dex ..  

## pvalue results Wald test p-value: d..  

## padj results BH adjusted p-values  

# Summary of DGE  

summary(dds_DGE_results)  

##  

## out of 22008 with nonzero total read count  

## adjusted p-value < 0.1  

## LFC > 0 (up) : 2624, 12%  

## LFC < 0 (down) : 2233, 10%  

## outliers [1] : 0, 0%  

## low counts [2] : 4267, 19%  

## (mean count < 5)  

## [1] see 'cooksCutoff' argument of ?results  

## [2] see 'independentFiltering' argument of ?results  

# Select significant genes with a p adjusted lower than 0.05  

resSig <- subset(dds_DGE_results, padj < 0.05)  

# Order significant genes by p adjusted  

resSig <- resSig[ order(resSig$padj), ]  

# Order significant genes with the strongest down-regulation  

head(resSig[ order(resSig$log2FoldChange), ])  

## log2 fold change (MLE): dex trt vs untrt  

## Wald test p-value: dex trt vs untrt  

## DataFrame with 6 rows and 6 columns  

## baseMean log2FoldChange lfcSE stat pvalue  

## <numeric> <numeric> <numeric> <numeric> <numeric>  

## ENSG00000128285 6.62454 -5.32504 1.248932 -4.26368 2.01091e-05  

## ENSG00000267339 26.23285 -4.61120 0.668404 -6.89882 5.24356e-12  

## ENSG00000019186 14.08722 -4.32425 0.850515 -5.08427 3.69048e-07  

## ENSG00000183454 5.80405 -4.26288 1.157619 -3.68246 2.30997e-04  

## ENSG00000146006 46.80589 -4.20876 0.525353 -8.01131 1.13496e-15  

## ENSG00000141469 53.43499 -4.12475 1.129823 -3.65079 2.61436e-04  

## padj  

## <numeric>  

## ENSG00000128285 2.06575e-04  

## ENSG00000267339 1.47895e-10  

## ENSG00000019186 5.36035e-06

```

```

## ENSG00000183454 1.82058e-03
## ENSG00000146006 4.89910e-14
## ENSG00000141469 2.03427e-03

# And with the strongest up-regulation
head(resSig[ order(resSig$log2FoldChange, decreasing = TRUE), ])

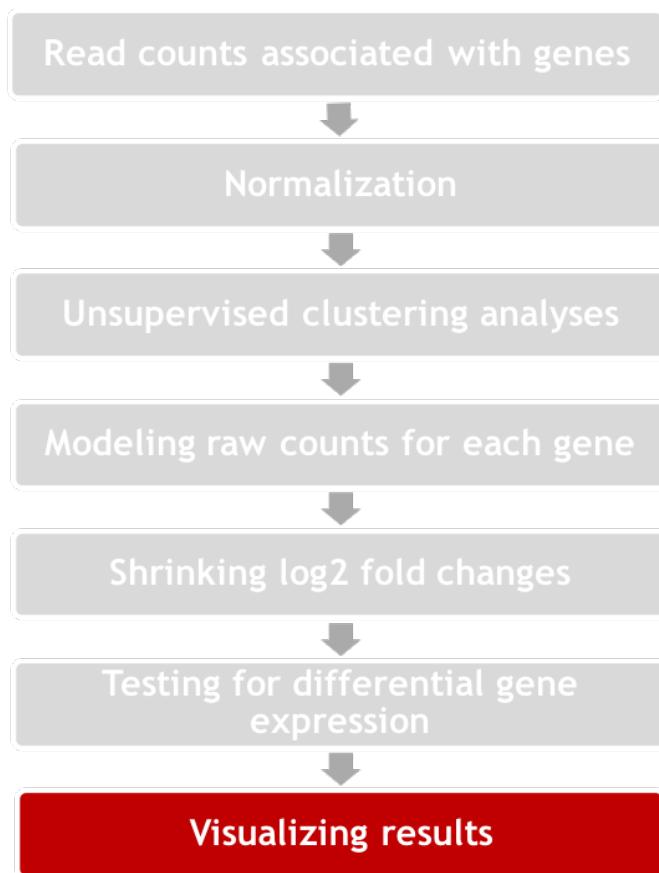
## log2 fold change (MLE): dex trt vs untrt
## Wald test p-value: dex trt vs untrt
## DataFrame with 6 rows and 6 columns
##           baseMean log2FoldChange      lfcSE      stat      pvalue
## <numeric>     <numeric> <numeric> <numeric> <numeric>
## ENSG00000179593 67.24111    9.50598  1.053319  9.02479 1.80039e-19
## ENSG00000109906 385.05722   7.35264  0.536394 13.70755 9.14995e-43
## ENSG00000250978 56.31694    6.32629  0.675580  9.36424 7.66013e-21
## ENSG00000132518 5.65448     5.88491  1.312168  4.48488 7.29565e-06
## ENSG00000127954 286.39779   5.20720  0.493144 10.55919 4.60624e-26
## ENSG00000249364 8.83902    5.09043  1.150576  4.42424 9.67817e-06
##          padj
## <numeric>
## ENSG00000179593 1.12864e-17
## ENSG00000109906 2.28633e-40
## ENSG00000250978 5.45777e-19
## ENSG00000132518 8.26515e-05
## ENSG00000127954 5.04440e-24
## ENSG00000249364 1.06978e-04

# Save significant DGE results to file
write.table(resSig, file = "DESeq2_Sig_results_dex_Vs_untrt.tab", sep = "\t", quote = FALSE, row.names

```

Exploratory plots

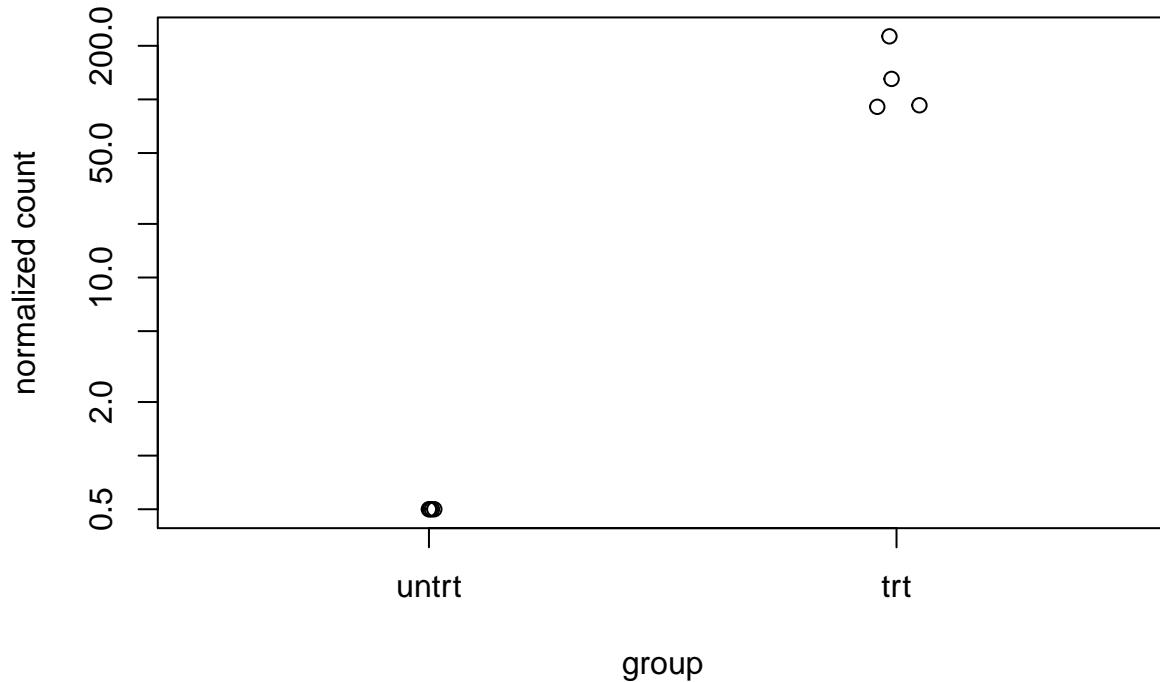
Plots are the best tools to explain DGE and your report should include them selecting those that best describe your findings.



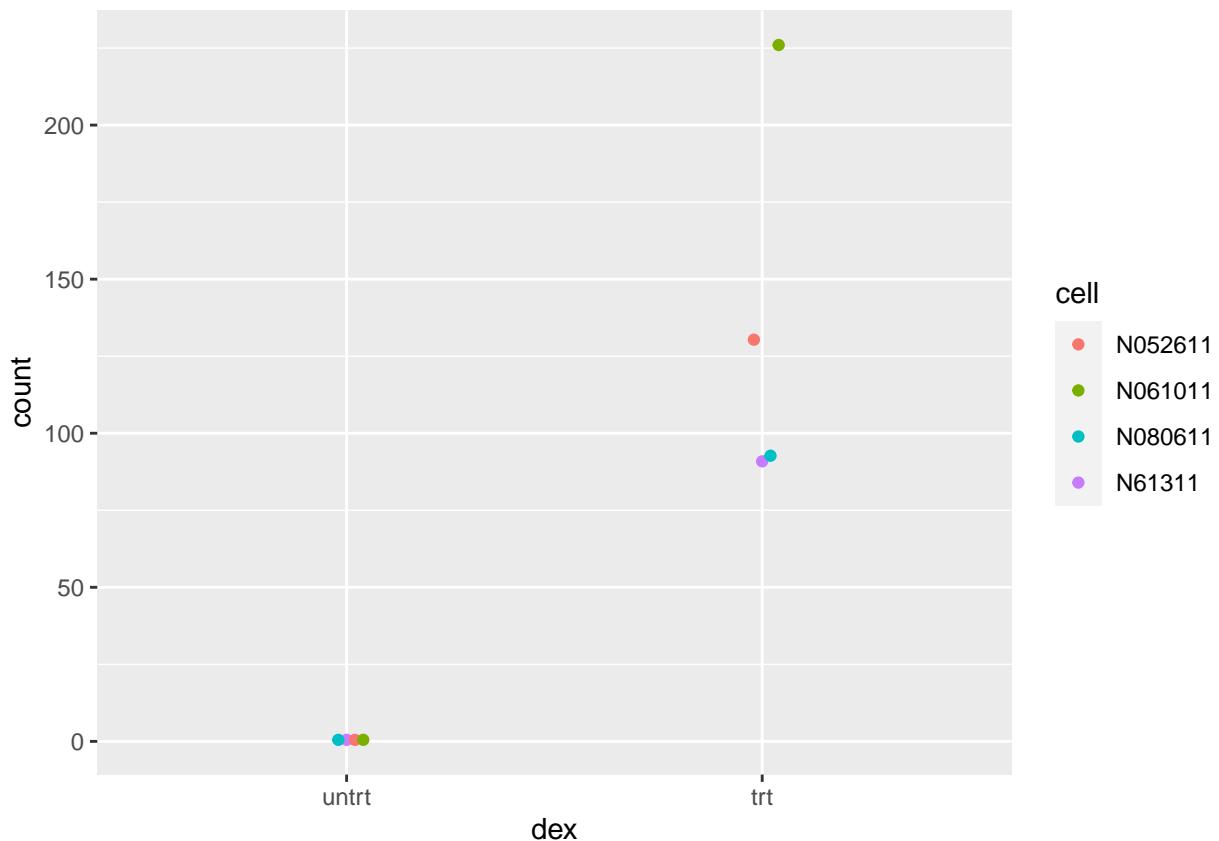
Individual gene count plot A quick way to visualize the counts for a particular gene is to use the `plotCounts` function that takes as arguments the `DESeqDataSet`, a gene name, and the group over which to plot the counts (figure below).

```
# Individual gene raw count plot
plotCounts(dds_DGE, gene = "ENSG00000179593", intgroup=c("dex"))
```

ENSG00000179593

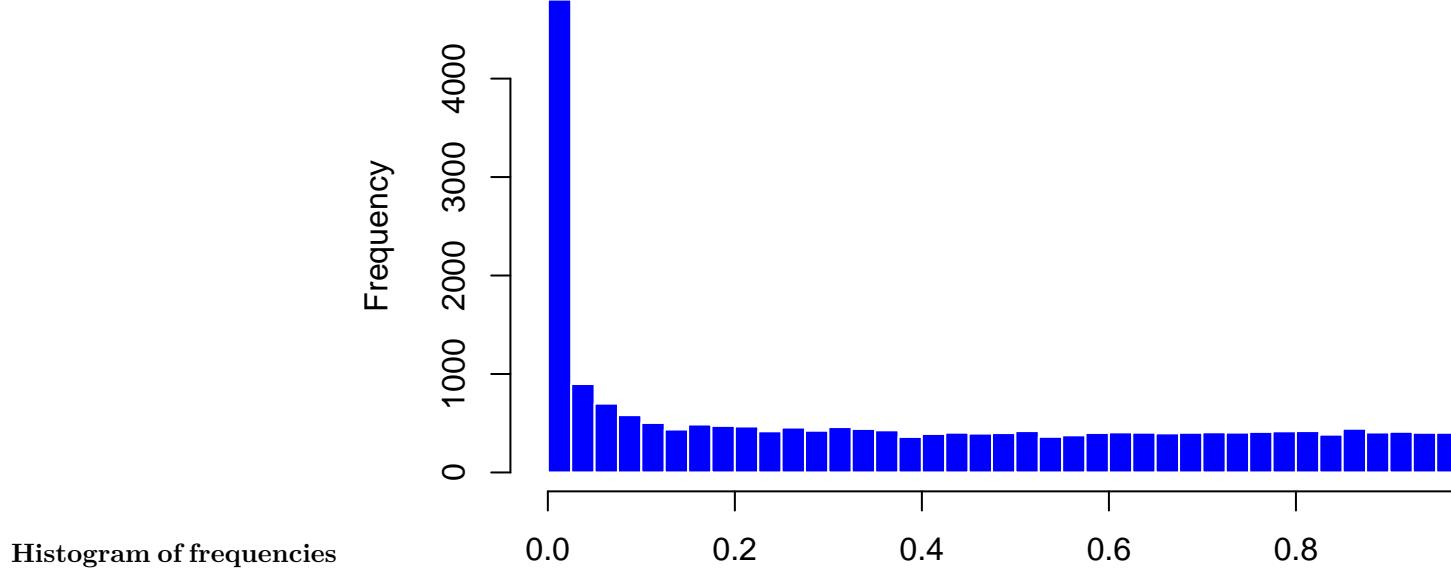


```
# Normalized counts for a single gene over treatment group.  
# We can also make custom plots using the ggplot function from the ggplot2 package (figures below).  
  
# BiocManager::install("ggbeeswarm")  
library("ggbeeswarm")  
geneCounts <- plotCounts(dds_DGE, gene = "ENSG00000179593", intgroup = c("dex","cell"),  
                         returnData = TRUE)  
ggplot(geneCounts, aes(x = dex, y = count, color = cell)) +  
  geom_beeswarm(cex = 1)
```



```
par(mfrow=c(1,1))
# Histogram of p-values frequencies
hist(dds_DGE_results$pvalue , col = "blue", xlab = "", border = "white", ylab = "Frequency", breaks =
```

frequencies of p-values



MA plot An MA-plot (Dudoit et al. 2002) provides a useful overview for the distribution of the estimated coefficients in the model, e.g. the comparisons of interest, across all genes. On the y-axis, the “M” stands for “minus” – subtraction of log values is equivalent to the log of the ratio – and on the x-axis, the “A” stands for “average”. This plot allows us to evaluate fold changes and the distribution around the mean expression.

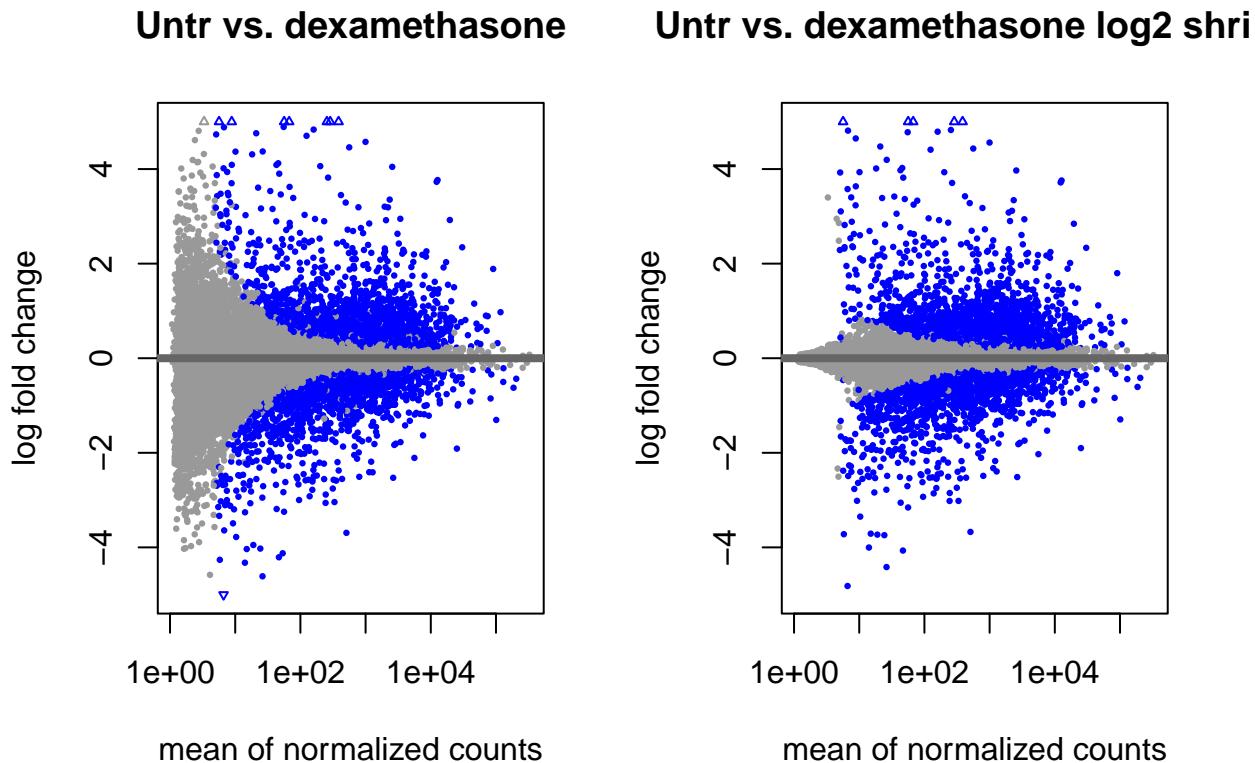
Before making the MA-plot, we use the lfcShrink function to shrink the log2 fold changes for the comparison of dex treated vs untreated samples. There are three types of shrinkage estimators in DESeq2, which are covered in the DESeq2 vignette. log2 fold shrinkage remove the noise preserving the large differences.

```
par(mfrow=c(1,2))
# MA plot without shrinkage
plotMA(dds_DGE_results, alpha = 0.05, main = "Untr vs. dexamethasone", ylim = c(-5,5))

# Log^2^ fold shrinkage
# BiocManager::install("apeglm")
# if (!require(apeglm)) {
#   install.packages("apeglm")
#   require(apeglm)
# }
library("apeglm")
# Check possible contrast to create the MA plot
resultsNames(dds_DGE)

## [1] "Intercept"           "cell_N061011_vs_N052611"
## [3] "cell_N080611_vs_N052611" "cell_N61311_vs_N052611"
## [5] "dex_trt_vs_untrt"

res <- lfcShrink(dds_DGE, coef="dex_trt_vs_untrt", type="apeglm")
plotMA(res, alpha = 0.05, main = "Untr vs. dexamethasone log2 shrink", ylim = c(-5, 5))
```



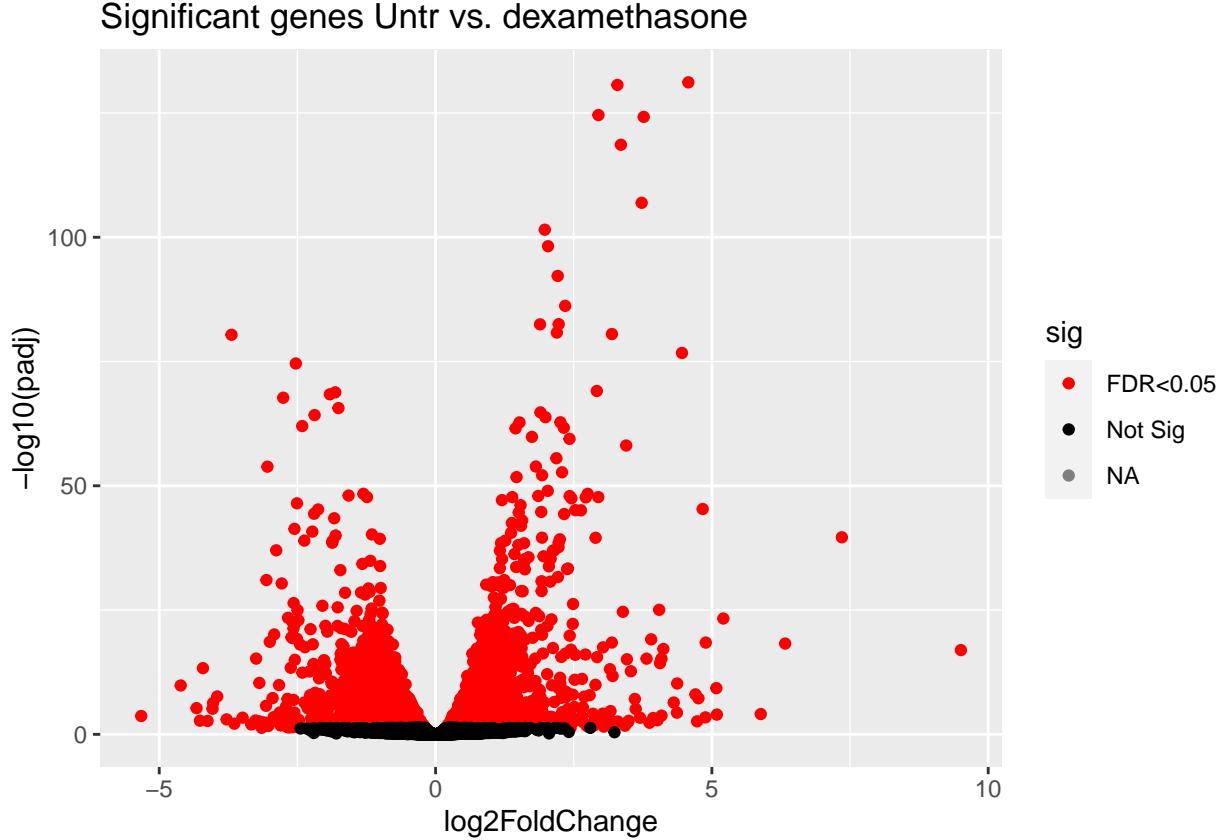
Volcano plot Volcano plot helps to visualize significant differential expressed genes showing the \log^2 fold change and the significance (p-adjusted).

```
# BiocManager::install("dplyr")
# if (!require(dplyr)) {
#   install.packages("dplyr")
#   require(dplyr)
# }
library("dplyr")

dds_DGE_results <- dds_DGE_results[order(dds_DGE_results$padj),]
results_order <- as.data.frame(dplyr::mutate(as.data.frame(dds_DGE_results), sig=ifelse(dds_DGE_results$padj < 0.05, "FDR<0.05", "Not Sig"), NA))

p = ggplot2::ggplot(results_order, ggplot2::aes(log2FoldChange, -log10(padj))) +
  ggplot2::geom_point(ggplot2::aes(col = sig)) +
  ggplot2::scale_color_manual(values = c("red", "black")) +
  ggplot2::ggtitle("Significant genes Untr vs. dexamethasone")

p
```



```
# Get ENSEMBL gene names
# Ggrepel library helps to no overlap gene names
# BiocManager::install("ggrepel")
# if (!require(ggrepel)) {
#   install.packages("ggrepel")
#   require(ggrepel)
# }
```

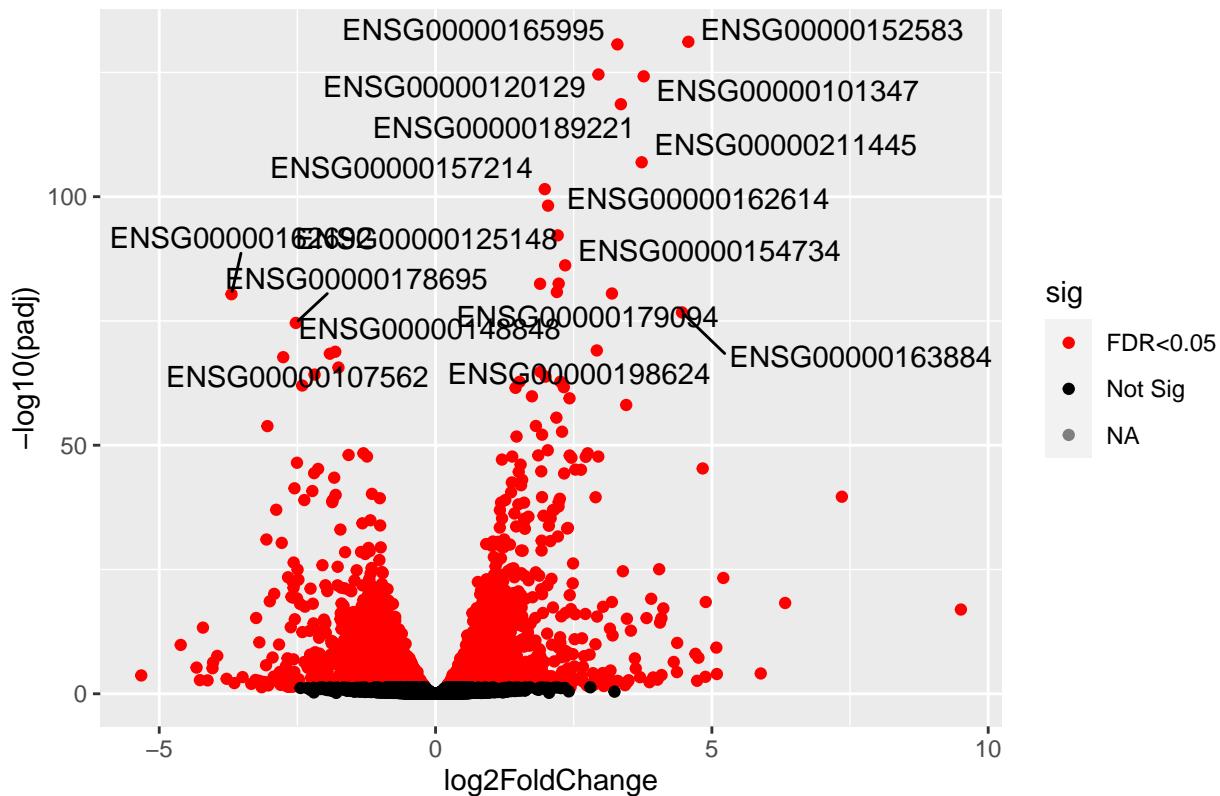
```

library("ggrepel")
# Select genes that have log2 fold change higher than 2 and p adjusted lower than 0.05
DEgenes_DESeq <- results_order[which(abs(results_order$log2FoldChange) > log2(2) & results_order$padj <
DEgenes_DESeq <- DEgenes_DESeq[order(DEgenes_DESeq$padj),]

# Plot adding
p + ggrepel::geom_text_repel(data=DEgenes_DESeq[1:20, ], ggplot2::aes(label=rownames(results_order)[1:20])

```

Significant genes Untr vs. dexamethasone

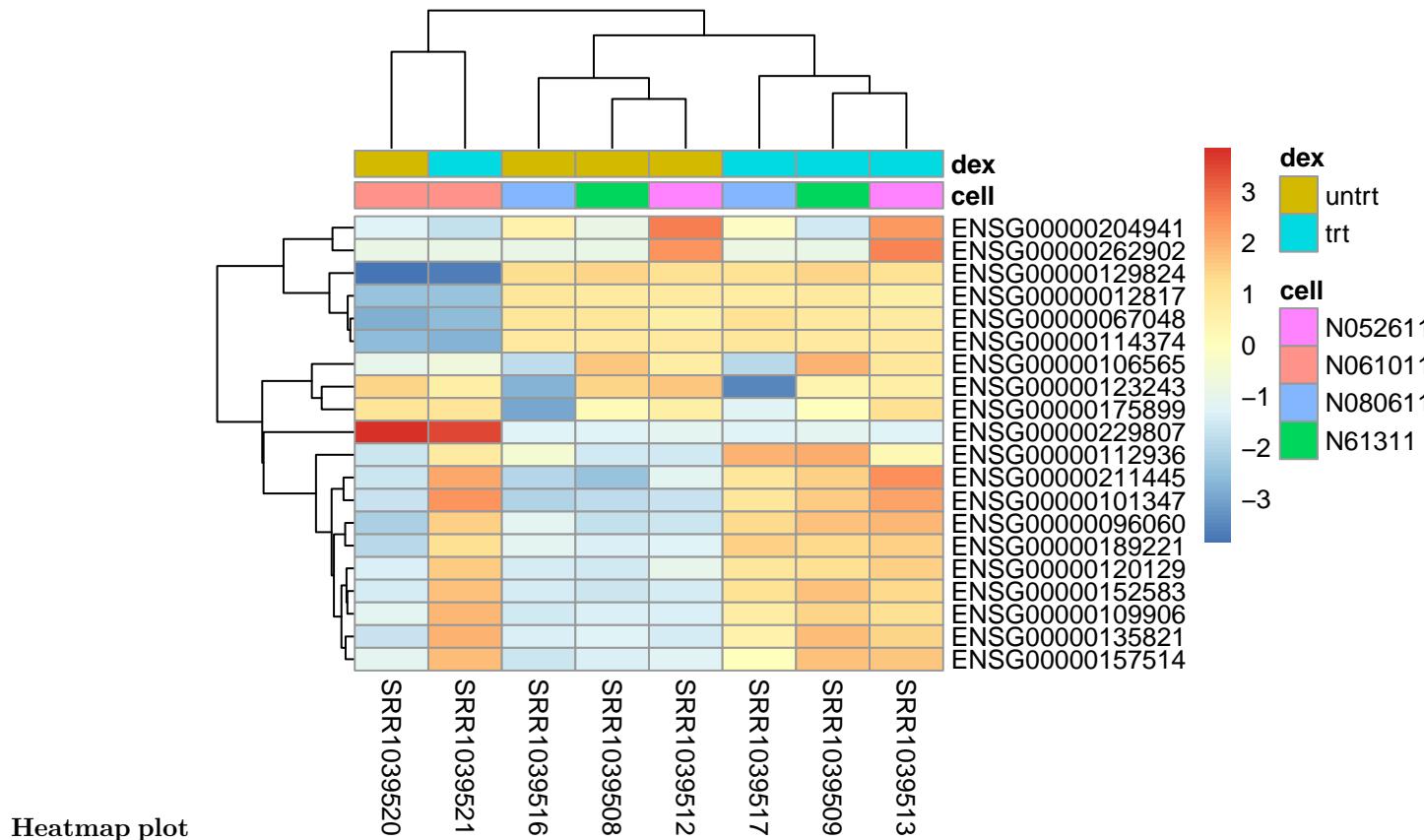


```

# Select the 20 most variable genes
topVarGenes <- head(order(rowVars(assay(vsd))), decreasing = TRUE), 20)

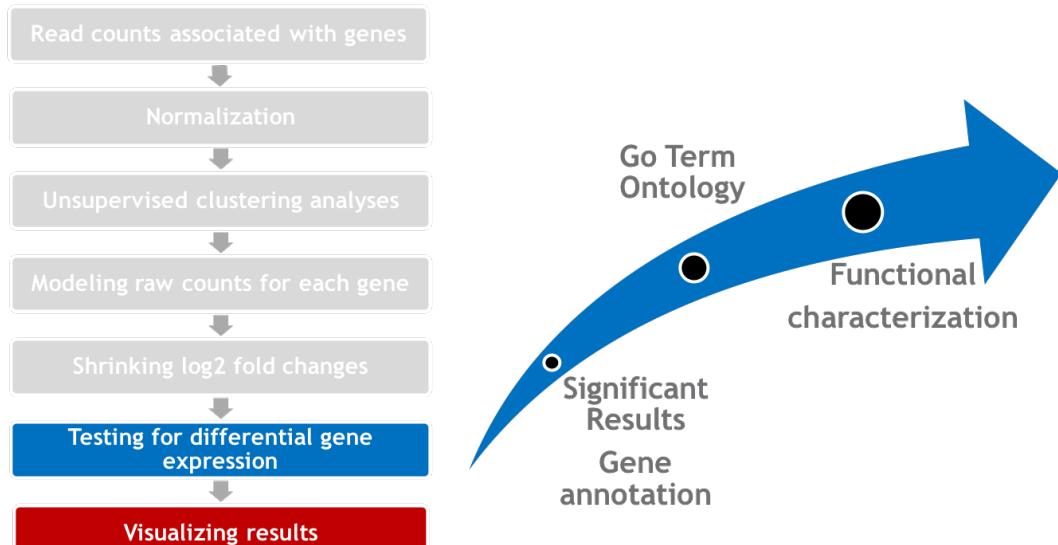
# Transform to matrix and use vsd transformation
mat <- assay(vsd)[ topVarGenes, ]
mat <- mat - rowMeans(mat)
anno <- as.data.frame(colData(vsd) [, c("cell","dex")])
pheatmap(mat, annotation_col = anno)

```



3. Annotate results

Our results data frame contains only ENSEMBL gene ID, but we need some more meaningful info. For that purpose we will add gene symbol, gene name, KEGG path and ENTREZ ID.



```

# Load AnnotationDbi package https://bioconductor.org/packages/3.9/AnnotationDbi
# BiocManager::install("AnnotationDbi")
# if (!require(AnnotationDbi)) {

```

```

#      install.packages("AnnotationDbi")
#      require(AnnotationDbi)
# }
library("AnnotationDbi")

# Load org.Hs.eg.db package https://bioconductor.org/packages/3.9/org.Hs.eg.db
# BiocManager::install("org.Hs.eg.db")
# if (!require(org.Hs.eg.db)) {
#   install.packages("org.Hs.eg.db")
#   require(org.Hs.eg.db)
# }
library("org.Hs.eg.db")

# Check all the possible terms that could be retrieved with AnnotationDbi "mapIds function"
columns(org.Hs.eg.db)

## [1] "ACCNUM"        "ALIAS"          "ENSEMBL"         "ENSEMLPROT"     "ENSEMLTRANS"
## [6] "ENTREZID"       "ENZYME"         "EVIDENCE"        "EVIDENCEALL"    "GENENAME"
## [11] "GENETYPE"       "GO"              "GOALL"           "IPI"             "MAP"
## [16] "OMIM"            "ONTOLOGY"        "ONTOLOGYALL"    "PATH"            "PFAM"
## [21] "PMID"           "PROSITE"         "REFSEQ"          "SYMBOL"          "UCSCKG"
## [26] "UNIPROT"

# Ordered significant genes
head(resSig)

## log2 fold change (MLE): dex trt vs untrt
## Wald test p-value: dex trt vs untrt
## DataFrame with 6 rows and 6 columns
##           baseMean log2FoldChange      lfcSE      stat      pvalue
## <numeric>      <numeric> <numeric> <numeric> <numeric>
## ENSG00000152583  997.410    4.57495  0.184224  24.8337 3.87953e-136
## ENSG00000165995  495.089    3.29110  0.132933  24.7576 2.56857e-135
## ENSG00000120129 3408.961    2.94784  0.121952  24.1721 4.37074e-129
## ENSG00000101347 12703.198    3.76701  0.156145  24.1250 1.36567e-128
## ENSG00000189221  2341.724    3.35365  0.142252  23.5754 6.89933e-123
## ENSG00000211445 12285.646    3.73043  0.166538  22.3999 3.94527e-111
##           padj
## <numeric>
## ENSG00000152583 6.88268e-132
## ENSG00000165995 2.27845e-131
## ENSG00000120129 2.58471e-125
## ENSG00000101347 6.05707e-125
## ENSG00000189221 2.44802e-119
## ENSG00000211445 1.16655e-107

# Add gene symbol column
resSig$SYMBOL = mapIds(org.Hs.eg.db,
                      keys=rownames(resSig),
                      column="SYMBOL",
                      keytype="ENSEMBL",
                      multiVals="first")

# Add gene name column
resSig$GENENAME = mapIds(org.Hs.eg.db,

```

```

        keys=rownames(resSig),
        column="GENENAME",
        keytype="ENSEMBL",
        multiVals="first")

# Add functional path column
resSig$PATH = mapIds(org.Hs.eg.db,
                      keys=rownames(resSig),
                      column="PATH",
                      keytype="ENSEMBL",
                      multiVals="first")

# Add entrez ID column
resSig$ENTREZID = mapIds(org.Hs.eg.db,
                         keys=rownames(resSig),
                         column="ENTREZID",
                         keytype="ENSEMBL",
                         multiVals="first")

# Add ENSEMBL column
resSig$ENSEMBL <- rownames(resSig)

resSig_DF <- as.data.frame(resSig)
# Save annotated data frame
write.table(resSig_DF, file="Significant_DE_dex_trt_vs_untrt_metadata.tab"), row.names = F,col.names =

```

4. Functional Gene Ontology (GO) analysis

Harvard Chan Bioinformatics Core (HBC) - https://github.com/hbctraining/DGE_workshop/blob/master/lessons/09_functional_analysis.md

gProfiler - <http://biit.cs.ut.ee/gprofiler/index.cgi>

DAVID - <http://david.abcc.ncifcrf.gov/tools.jsp>

clusterProfiler - <http://bioconductor.org/packages/release/bioc/html/clusterProfiler.html>

```

# Load clusterProfiler package  http://bioconductor.org/packages/release/bioc/html/clusterProfiler.html
# BiocManager::install("clusterProfiler")
# if (!require(clusterProfiler)) {
#   install.packages("clusterProfiler")
#   require(clusterProfiler)
# }
library("clusterProfiler")

##
## clusterProfiler v4.10.0 For help: https://yulab-smu.top/biomedical-knowledge-mining-book/
##
## If you use clusterProfiler in published research, please cite:
## T Wu, E Hu, S Xu, M Chen, P Guo, Z Dai, T Feng, L Zhou, W Tang, L Zhan, X Fu, S Liu, X Bo, and G Yu.

##
## Attaching package: 'clusterProfiler'

## The following object is masked from 'package:AnnotationDbi':

```

```

##      select
## The following object is masked from 'package:IRanges':
##      slice
## The following object is masked from 'package:S4Vectors':
##      rename
## The following object is masked from 'package:stats':
##      filter
library("enrichplot")
# BiocManager::install("ggnewscale")
library("ggnewscale")

OrgDb <- org.Hs.eg.db # can also be other organisms
# Get ENTREZID as vector. genes list are gonna be used to call GO terms
genes <- as.character(resSig$ENTREZID)

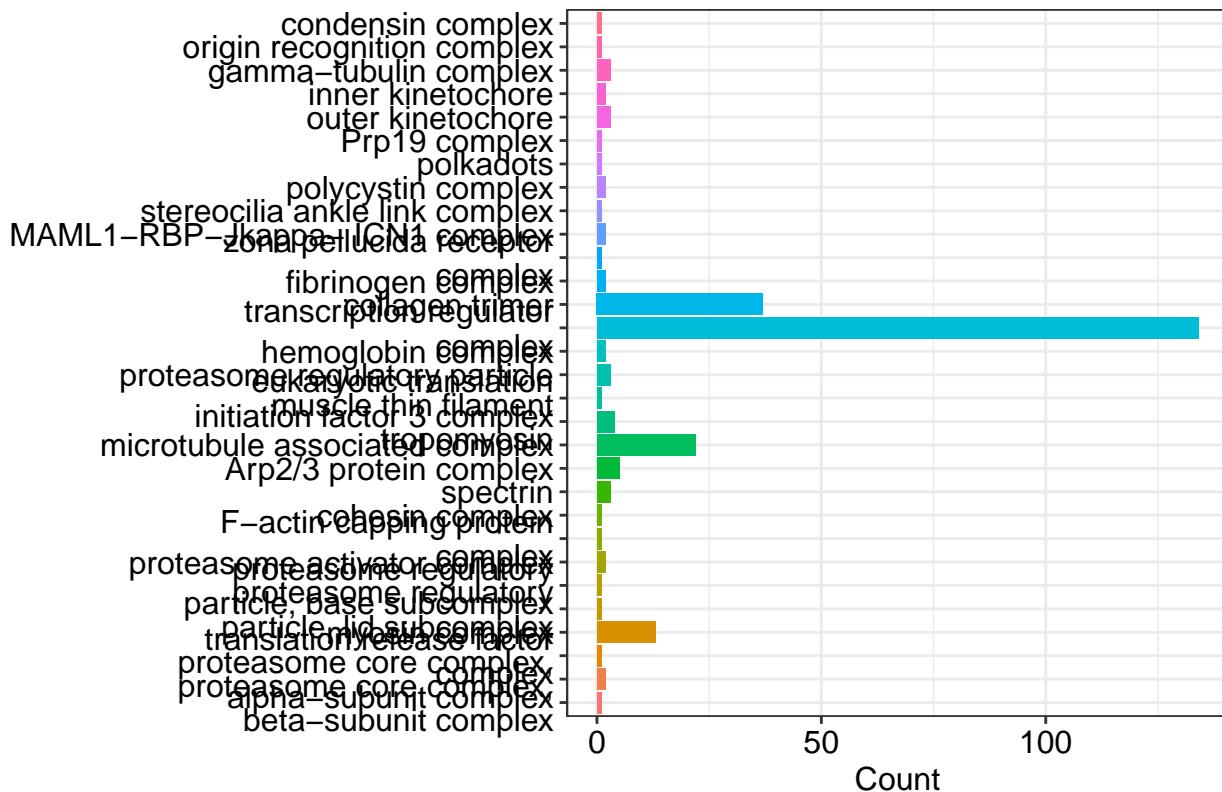
# GeneOntology terms

## Cellular compartment
# In clusterProfiler, groupGO is designed for gene classification based on GO distribution at a specific
ggo <- clusterProfiler::groupGO(gene      = genes,
                                OrgDb    = OrgDb,
                                ont      = "CC",      #MF , BP CC
                                level    = 3,
                                readable = TRUE)
head(as.data.frame(ggo) [,-5])

##           ID          Description Count GeneRatio
## GO:0000133 GO:0000133 polarisome      0   0/3774
## GO:0000408 GO:0000408 EKC/KEOPS complex  0   0/3774
## GO:0000417 GO:0000417 HIR complex      0   0/3774
## GO:0000796 GO:0000796 condensin complex  1   1/3774
## GO:0000808 GO:0000808 origin recognition complex 1   1/3774
## GO:0000930 GO:0000930 gamma-tubulin complex 3   3/3774

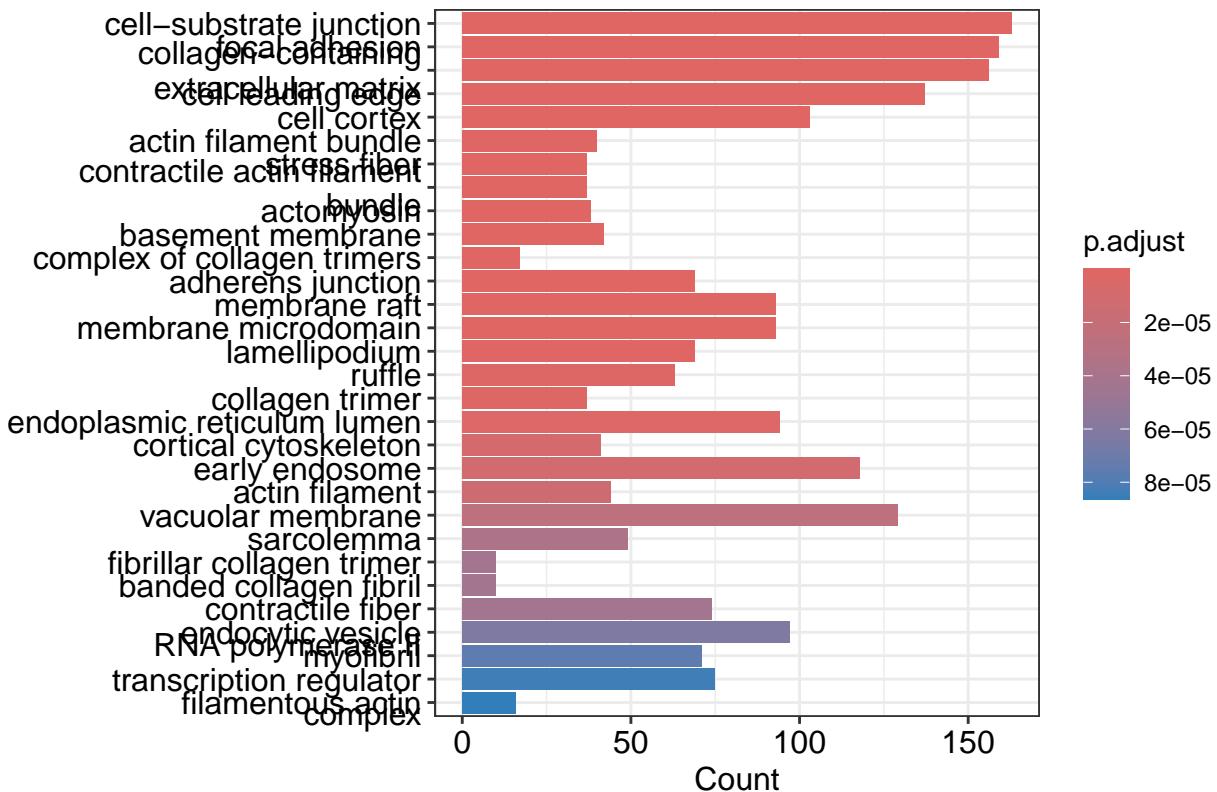
barplot(ggo, drop=TRUE, showCategory=30, vertex.label.cex=0.8)

```



```
# GO over-representation test
ego <- clusterProfiler::enrichGO(gene      = genes,
                                    OrgDb     = OrgDb,
                                    ont       = "CC",
                                    pAdjustMethod = "BH",
                                    pvalueCutoff = 0.01,
                                    qvalueCutoff = 0.01,
                                    readable   = TRUE)

barplot(ego, showCategory=30)
```



```

# Gene Concept Network
edox <- setReadable(ego, 'org.Hs.eg.db', 'ENTREZID')

## Gene list with Fold Change data
geneList <- resSig$log2FoldChange
names(geneList) <- as.character(unique(resSig$ENTREZID))
geneList <- sort(geneList, decreasing = TRUE)
#de <- names(geneList)[abs(geneList) > 2]
head(geneList)

##      11030     7704     6603    26135     84909     54456
## 9.505982 7.352645 6.326295 5.884912 5.207201 5.090427
## categorySize can be scaled by 'pvalue' or 'geneNum'
p1 <- cnetplot(edox, categorySize="pvalue", foldChange= geneList)

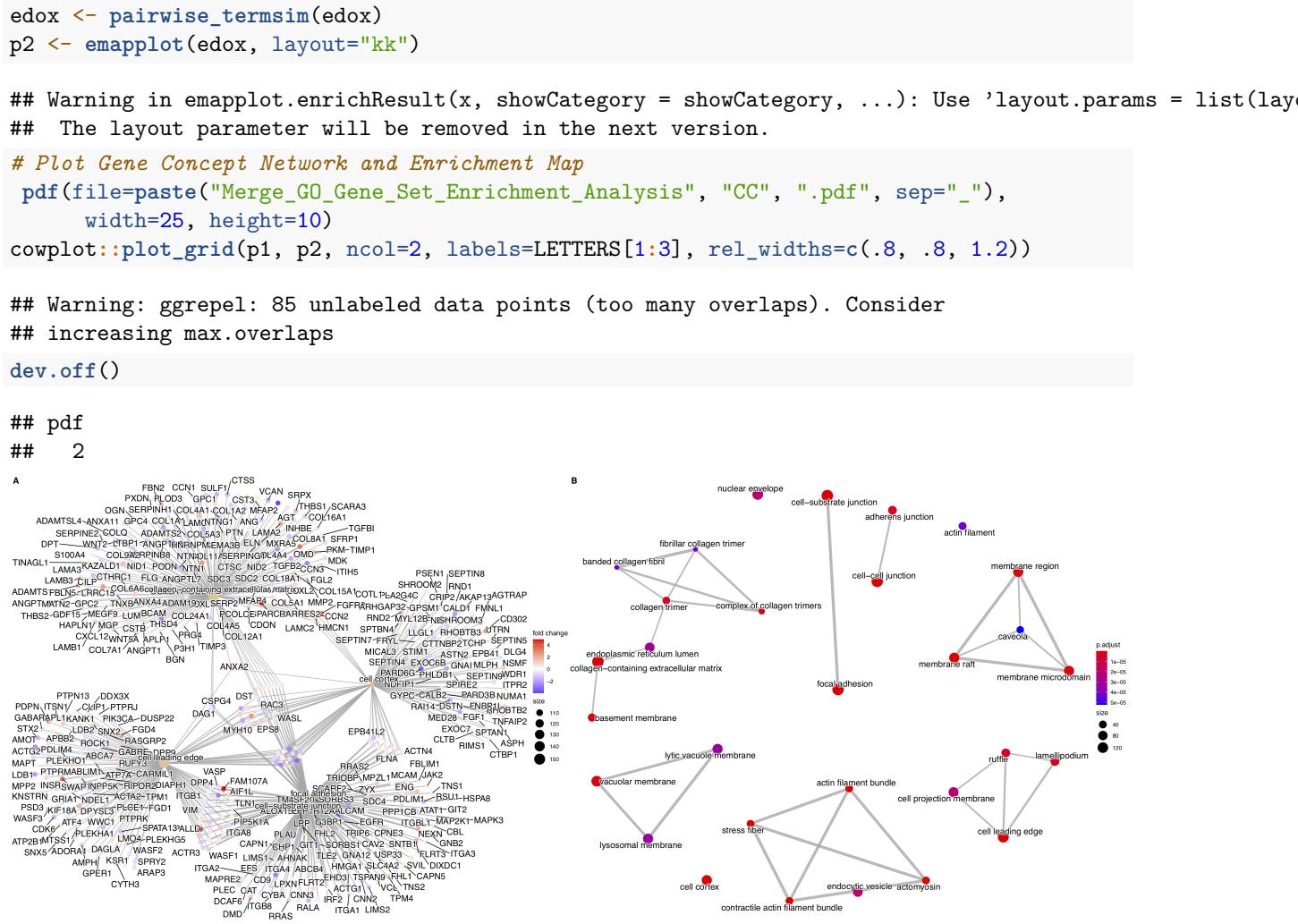
## Warning in cnetplot.enrichResult(x, ...): Use 'color.params = list(foldChange = your_value)' instead
## The foldChange parameter will be removed in the next version.

## Scale for size is already present.
## Adding another scale for size, which will replace the existing scale.

# Enrichment Map
# Enrichment map organizes enriched terms into a network with edges connecting overlapping gene sets.
# In this way, mutually overlapping gene sets are tend to cluster together, making it easy to identify .
library(DOSE)

## DOSE v3.28.2 For help: https://yulab-smu.top/biomedical-knowledge-mining-book/
##
## If you use DOSE in published research, please cite:
## Guangchuang Yu, Li-Gen Wang, Guang-Rong Yan, Qing-Yu He. DOSE: an R/Bioconductor package for Disease

```



References

- Anders, Simon, and Wolfgang Huber. 2010. “Differential expression analysis for sequence count data.” *Genome Biology* 11 (10):R106+. <https://doi.org/10.1186/gb-2010-11-10-r106>.
- Dudoit, Rine, Yee H. Yang, Matthew J. Callow, and Terence P. Speed. 2002. “Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments.” *Statistica Sinica*, 111–39.
- Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. “Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2.” *Genome Biology* 15 (12). BioMed Central Ltd:550+. <https://doi.org/10.1186/s13059-014-0550-8>.
- Michael Love. RNA-seq workflow: gene-level exploratory analysis and differential expression. 2018 ** <https://www.bioconductor.org/packages/devel/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html>
- Himes, Blanca E., Xiaofeng Jiang, Peter Wagner, Ruoxi Hu, Qiyu Wang, Barbara Klanderman, Reid M. Whitaker, et al. 2014. “RNA-Seq transcriptome profiling identifies CRISPLD2 as a glucocorticoid responsive gene that modulates cytokine function in airway smooth muscle cells.” *PloS One* 9 (6). <https://doi.org/10.1371/journal.pone.0099625>.

Session Info

We should always keep record of the R packages used and their versions. In some cases you could get different results with different packages version!!

```
devtools::session_info()
```

```
## - Session info -
## setting value
## version R version 4.3.2 (2023-10-31)
## os      Ubuntu 22.04.3 LTS
## system x86_64, linux-gnu
## ui      X11
## language (EN)
## collate en_US.UTF-8
## ctype   en_US.UTF-8
## tz      Etc/UTC
## date    2024-02-21
## pandoc 3.1.1 @ /usr/lib/rstudio-server/bin/quarto/bin/tools/ (via rmarkdown)
##
## - Packages -
## package           * version    date (UTC) lib source
## abind              1.4-5       2016-07-21 [1] RSPM (R 4.3.0)
## affy               1.80.0      2023-10-24 [1] Bioconductor
## affyio              1.72.0      2023-10-24 [1] Bioconductor
## airway              * 1.22.0      2023-10-26 [1] Bioconductor
## AnnotationDbi      * 1.64.1      2023-11-03 [1] Bioconductor
## ape                 5.7-1       2023-03-13 [1] RSPM (R 4.3.0)
## apeglm              * 1.24.0      2023-10-24 [1] Bioconductor
## aplot                0.2.2       2023-10-06 [1] RSPM (R 4.3.0)
## bbmle                1.0.25.1    2023-12-09 [1] RSPM (R 4.3.0)
## bdsmatrix             1.3-6       2022-06-03 [1] RSPM (R 4.3.0)
## beeswarm              0.4.0       2021-06-01 [1] RSPM (R 4.3.0)
## Biobase              * 2.62.0      2023-10-24 [1] Bioconductor
## BiocGenerics          * 0.48.1      2023-11-01 [1] Bioconductor
## BiocManager            1.30.22     2023-08-08 [1] RSPM (R 4.3.0)
## BiocParallel            1.36.0      2023-10-24 [1] Bioconductor
## Biostrings             2.70.2      2024-01-28 [1] Bioconductor 3.18 (R 4.3.2)
## bit                  4.0.5       2022-11-15 [1] RSPM (R 4.3.0)
## bit64                 4.0.5       2020-08-30 [1] RSPM (R 4.3.0)
## bitops                 1.0-7       2021-04-24 [1] RSPM (R 4.3.0)
## blob                  1.2.4       2023-03-17 [1] RSPM (R 4.3.0)
## cachem                 1.0.8       2023-05-01 [1] RSPM (R 4.3.0)
## cli                   3.6.2       2023-12-11 [1] RSPM (R 4.3.0)
## clusterProfiler        * 4.10.0      2023-10-24 [1] Bioconductor
## coda                  0.19-4.1    2024-01-31 [1] RSPM (R 4.3.0)
## codetools              0.2-19      2023-02-01 [2] CRAN (R 4.3.2)
## colorspace              2.1-0       2023-01-23 [1] RSPM (R 4.3.0)
## cowplot                 1.1.3       2024-01-22 [1] RSPM (R 4.3.0)
## crayon                  1.5.2       2022-09-29 [1] RSPM (R 4.3.0)
## data.table              1.15.0      2024-01-30 [1] RSPM (R 4.3.0)
## DBI                   1.2.2       2024-02-16 [1] RSPM (R 4.3.0)
## DelayedArray             0.28.0      2023-10-24 [1] Bioconductor
## DESeq2                 * 1.42.0      2023-10-24 [1] Bioconductor
## devtools                 2.4.5       2022-10-11 [1] RSPM (R 4.3.0)
```

## digest	0.6.34	2024-01-11 [1] RSPM (R 4.3.0)
## DOSE	* 3.28.2	2023-12-10 [1] Bioconductor 3.18 (R 4.3.2)
## dplyr	* 1.1.4	2023-11-17 [1] RSPM (R 4.3.0)
## ellipsis	0.3.2	2021-04-29 [1] RSPM (R 4.3.0)
## emdbook	1.3.13	2023-07-03 [1] RSPM (R 4.3.0)
## enrichplot	* 1.22.0	2023-10-24 [1] Bioconductor
## evaluate	0.23	2023-11-01 [1] RSPM (R 4.3.0)
## fansi	1.0.6	2023-12-08 [1] RSPM (R 4.3.0)
## farver	2.1.1	2022-07-06 [1] RSPM (R 4.3.0)
## fastmap	1.1.1	2023-02-24 [1] RSPM (R 4.3.0)
## fastmatch	1.1-4	2023-08-18 [1] RSPM (R 4.3.0)
## fgsea	1.28.0	2023-10-24 [1] Bioconductor
## fs	1.6.3	2023-07-20 [1] RSPM (R 4.3.0)
## generics	0.1.3	2022-07-05 [1] RSPM (R 4.3.0)
## GenomeInfoDb	* 1.38.6	2024-02-08 [1] Bioconductor 3.18 (R 4.3.2)
## GenomeInfoDbData	1.2.11	2024-02-20 [1] Bioconductor
## GenomicRanges	* 1.54.1	2023-10-29 [1] Bioconductor
## ggbeeswarm	* 0.7.2	2023-04-29 [1] RSPM (R 4.3.0)
## ggforce	0.4.1	2022-10-04 [1] RSPM (R 4.3.0)
## ggfun	0.1.4	2024-01-19 [1] RSPM (R 4.3.0)
## ggnewscale	* 0.4.10	2024-02-08 [1] RSPM (R 4.3.0)
## ggplot2	* 3.4.4	2023-10-12 [1] RSPM (R 4.3.0)
## ggplotify	0.1.2	2023-08-09 [1] RSPM (R 4.3.0)
## ggraph	2.1.0	2022-10-09 [1] RSPM (R 4.3.0)
## ggrepel	* 0.9.5	2024-01-10 [1] RSPM (R 4.3.0)
## ggtree	3.10.0	2023-10-24 [1] Bioconductor
## glue	1.7.0	2024-01-09 [1] RSPM (R 4.3.0)
## GO.db	3.18.0	2024-02-20 [1] Bioconductor
## GOSemSim	2.28.1	2024-01-17 [1] Bioconductor 3.18 (R 4.3.2)
## graphlayouts	1.1.0	2024-01-19 [1] RSPM (R 4.3.0)
## gridExtra	2.3	2017-09-09 [1] RSPM (R 4.3.0)
## gridGraphics	0.5-1	2020-12-13 [1] RSPM (R 4.3.0)
## gson	0.1.0	2023-03-07 [1] RSPM (R 4.3.0)
## gtable	0.3.4	2023-08-21 [1] RSPM (R 4.3.0)
## HDO.db	0.99.1	2024-02-20 [1] Bioconductor
## hexbin	* 1.28.3	2023-03-21 [1] RSPM (R 4.3.0)
## highr	0.10	2022-12-22 [1] RSPM (R 4.3.0)
## htmltools	0.5.7	2023-11-03 [1] RSPM (R 4.3.0)
## htmlwidgets	1.6.4	2023-12-06 [1] RSPM (R 4.3.0)
## httpuv	1.6.14	2024-01-26 [1] RSPM (R 4.3.0)
## httr	1.4.7	2023-08-15 [1] RSPM (R 4.3.0)
## igraph	2.0.2	2024-02-17 [1] RSPM (R 4.3.0)
## IRanges	* 2.36.0	2023-10-24 [1] Bioconductor
## jsonlite	1.8.8	2023-12-04 [1] RSPM (R 4.3.0)
## KEGGREST	1.42.0	2023-10-24 [1] Bioconductor
## knitr	1.45	2023-10-30 [1] RSPM (R 4.3.0)
## labeling	0.4.3	2023-08-29 [1] RSPM (R 4.3.0)
## later	1.3.2	2023-12-06 [1] RSPM (R 4.3.0)
## lattice	0.22-5	2023-10-24 [2] RSPM (R 4.3.0)
## lazyeval	0.2.2	2019-03-15 [1] RSPM (R 4.3.0)
## lifecycle	1.0.4	2023-11-07 [1] RSPM (R 4.3.0)
## limma	3.58.1	2023-10-31 [1] Bioconductor
## locfit	1.5-9.8	2023-06-11 [1] RSPM (R 4.3.0)
## magrittr	2.0.3	2022-03-30 [1] RSPM (R 4.3.0)

```

## MASS           7.3-60.0.1 2024-01-13 [2] RSPM (R 4.3.0)
## Matrix         1.6-5     2024-01-11 [2] RSPM (R 4.3.0)
## MatrixGenerics * 1.14.0   2023-10-24 [1] Bioconductor
## matrixStats    * 1.2.0    2023-12-11 [1] RSPM (R 4.3.0)
## memoise        2.0.1    2021-11-26 [1] RSPM (R 4.3.0)
## mime           0.12     2021-09-28 [1] RSPM (R 4.3.0)
## miniUI         0.1.1.1   2018-05-18 [1] RSPM (R 4.3.0)
## munsell        0.5.0    2018-06-12 [1] RSPM (R 4.3.0)
## mvtnorm        1.2-4     2023-11-27 [1] RSPM (R 4.3.0)
## nlme           3.1-164   2023-11-27 [2] RSPM (R 4.3.0)
## numDeriv       2016.8-1.1 2019-06-06 [1] RSPM (R 4.3.0)
## org.Hs.eg.db   * 3.18.0   2024-02-20 [1] Bioconductor
## patchwork      1.2.0    2024-01-08 [1] RSPM (R 4.3.0)
## pheatmap       * 1.0.12   2019-01-04 [1] RSPM (R 4.3.0)
## pillar          1.9.0    2023-03-22 [1] RSPM (R 4.3.0)
## pkgbuild       1.4.3     2023-12-10 [1] RSPM (R 4.3.0)
## pkgconfig       2.0.3     2019-09-22 [1] RSPM (R 4.3.0)
## pkgload         1.3.4     2024-01-16 [1] RSPM (R 4.3.0)
## plyr            1.8.9     2023-10-02 [1] RSPM (R 4.3.0)
## png              0.1-8     2022-11-29 [1] RSPM (R 4.3.0)
## polyclip        1.10-6    2023-09-27 [1] RSPM (R 4.3.0)
## preprocessCore  1.64.0    2023-10-24 [1] Bioconductor
## profvis         0.3.8     2023-05-02 [1] RSPM (R 4.3.0)
## promises        1.2.1     2023-08-10 [1] RSPM (R 4.3.0)
## purrr           1.0.2     2023-08-10 [1] RSPM (R 4.3.0)
## qvalue          2.34.0    2023-10-24 [1] Bioconductor
## R6               2.5.1     2021-08-19 [1] RSPM (R 4.3.0)
## RColorBrewer    * 1.1-3     2022-04-03 [1] RSPM (R 4.3.0)
## Rcpp             1.0.12    2024-01-09 [1] RSPM (R 4.3.0)
## RCurl            1.98-1.14  2024-01-09 [1] RSPM (R 4.3.0)
## remotes          2.4.2.1   2023-07-18 [1] RSPM (R 4.3.0)
## reshape2         1.4.4     2020-04-09 [1] RSPM (R 4.3.0)
## rlang            1.1.3     2024-01-10 [1] RSPM (R 4.3.0)
## rmarkdown         2.25      2023-09-18 [1] RSPM (R 4.3.0)
## RSQLite          2.3.5     2024-01-21 [1] RSPM (R 4.3.0)
## rstudioapi       0.15.0    2023-07-07 [1] RSPM (R 4.3.0)
## S4Arrays         1.2.0     2023-10-24 [1] Bioconductor
## S4Vectors        * 0.40.2   2023-11-23 [1] Bioconductor 3.18 (R 4.3.2)
## scales           1.3.0     2023-11-28 [1] RSPM (R 4.3.0)
## scatterpie       0.2.1     2023-06-07 [1] RSPM (R 4.3.0)
## sessioninfo      1.2.2     2021-12-06 [1] RSPM (R 4.3.0)
## shadowtext       0.1.3     2024-01-19 [1] RSPM (R 4.3.0)
## shiny             1.8.0     2023-11-17 [1] RSPM (R 4.3.0)
## SparseArray      1.2.4     2024-02-11 [1] Bioconductor 3.18 (R 4.3.2)
## statmod          1.5.0     2023-01-06 [1] RSPM (R 4.3.0)
## stringi           1.8.3     2023-12-11 [1] RSPM (R 4.3.0)
## stringr           1.5.1     2023-11-14 [1] RSPM (R 4.3.0)
## SummarizedExperiment * 1.32.0  2023-10-24 [1] Bioconductor
## tibble            3.2.1     2023-03-20 [1] RSPM (R 4.3.0)
## tidygraph          1.3.1     2024-01-30 [1] RSPM (R 4.3.0)
## tidyrr             1.3.1     2024-01-24 [1] RSPM (R 4.3.0)
## tidyselect         1.2.0     2022-10-10 [1] RSPM (R 4.3.0)
## tidytree           0.4.6     2023-12-12 [1] RSPM (R 4.3.0)
## treeio            1.26.0    2023-10-24 [1] Bioconductor

```

```
##  tweenr           2.0.2    2022-09-06 [1] RSPM (R 4.3.0)
##  urlchecker       1.0.1    2021-11-30 [1] RSPM (R 4.3.0)
##  usethis          2.2.2    2023-07-06 [1] RSPM (R 4.3.0)
##  utf8             1.2.4    2023-10-22 [1] RSPM (R 4.3.0)
##  vctrs             0.6.5    2023-12-01 [1] RSPM (R 4.3.0)
##  viper            0.4.7    2023-12-18 [1] RSPM (R 4.3.0)
##  viridis           0.6.5    2024-01-29 [1] RSPM (R 4.3.0)
##  viridisLite      0.4.2    2023-05-02 [1] RSPM (R 4.3.0)
##  vsn              * 3.70.0   2023-10-24 [1] Bioconductor
##  withr            3.0.0    2024-01-16 [1] RSPM (R 4.3.0)
##  xfun              0.42     2024-02-08 [1] RSPM (R 4.3.0)
##  xtable            1.8-4    2019-04-21 [1] RSPM (R 4.3.0)
##  XVector           0.42.0   2023-10-24 [1] Bioconductor
##  yaml              2.3.8    2023-12-11 [1] RSPM (R 4.3.0)
##  yulab.utils       0.1.4    2024-01-28 [1] RSPM (R 4.3.0)
##  zlibbioc          1.48.0   2023-10-24 [1] Bioconductor
##
##  [1] /usr/local/lib/R/site-library
##  [2] /usr/local/lib/R/library
##  -----
##
```