Workshop #6 Differential Gene Expression Analysis

Yidong Chen

October 4, 2020

```
# # packages you may need to install for this lecture.
# BiocManager::install( "phantasus" )
# BiocManager::install("EnhancedVolcano")
# BiocManager::install( "DESeq2" )
# BiocManager::install("airway")
# BiocManager::install("EnhancedVolcano")
# BiocManager::install("VennDiagram")
#
# # if you have not installed following packages in earlier lectures
# BiocManager::install( "org.Hs.eg.db" )
# BiocManager::install( "AnnotationDbi" )
# BiocManager::install( "dplyr" )
# install.packages("plotly")
```

Example 1: Interactive gene expression data analysis.

First example (using GSE53986) is mainly for the purpose of understanding the basic gene expression anaysis tasks:

1. data transformation,
2. normalization,
3. differential expression,
4. gene selction, and
5. clustering

Please join other workshops for alignment, quantification, as well as functional analsys, and public-domain data importing.

```
# First example R phantasus for interactive data analysis
# If need to install, uncomment the next line.
#BiocManager::install( "phantasus" )
suppressMessages( library( phantasus ) )

# start the server during demo.
# servePhantasus()
```

Example 2a: Using airway AND DESeq2.

Using DESeq for differential gene expression analysis.

```r
# Third example: airway & DESeq.
suppressMessages( library( airway ) )
suppressMessages( library("DESeq2") )
suppressMessages( library("ggplot2") )

# load data.
data( "airway" )
se <- airway

# We set up differentially expressed gene selection criterion here.
pThr        <- 0.01   # for adj. p-value threshold (<= 0.01)
logFCThr    <- 1      # for log2 fold-change (at least 2 fold)
baseMeanThr <- 20     # for average expression level (at least 20 counts).
cpmThr      <- 1      # for copy-per-million (at least 1 cpm).

# examine all data component
head( assay( se ) )
```

```
##                 SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003        679        448        873        408       1138
## ENSG00000000005          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
## ENSG00000000003       1047        770        572
## ENSG00000000005          0          0          0
## ENSG00000000419        799        417        508
## ENSG00000000457        331        233        229
## ENSG00000000460         63         76         60
## ENSG00000000938          0          0          0
```

```r
rowData( se )
```

```
## DataFrame with 64102 rows and 0 columns
```

```r
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
```

```
# create DESeq data set, using paired design.
dds <- DESeqDataSet(se, design = ~ cell + dex)

# you can also try just do treated vs untreated, without pairing samples.
#dds <- DESeqDataSet(se, design = ~dex )

# What about if we only have data matrix, and sample info, such as,
countMat <- assay( se )
sampleInfo <- as.data.frame( colData( se ) )
ddsMat <- DESeqDataSetFromMatrix( countData = countMat,
                                  colData = sampleInfo,
                                  design = ~ cell + dex )

# Perform differential gene expression analysis, and
dds <- DESeq( dds )
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
# get the result out.
res <- results( dds )

# select genes that are significant. DESeq2 uses the Benjamini-Hochberg (BH) adjustment as
# described in the base R p.adjust function. Note, this is the fraction of false positives
# (the false discovery rate, FDR).  BH-adjusted p values are given in the column padj of
# the res object.
idx = which( res$padj <= pThr &
             abs( res$log2FoldChange ) >= logFCThr &
             res$baseMean >= baseMeanThr )
sigRes = res[idx, ]

# write out to a file. Check the file and you may not like it.
write.table(sigRes, file = "airway_sigGenes_adjp0.05_lfc1_bm20.txt",
            sep = "\t", col.names = TRUE, row.names = TRUE)
```

Break #1: Other dataset.

Student may explore this website,
http://www.bioconductor.org/packages/release/data/experiment/,
and select a particular dataset to repeat the process. We recommend pasilla

```
if( ! require(pasilla) ) {
  BiocManager::install("pasilla")
  library("pasilla")
}
```

```
## Loading required package: pasilla
```

```
# pasilla data set contains two count matrices: per-gene read counts and
# per-exon read counts. Here we use per-gene read count matrix.
data("pasillaGenes")
```

```
## Warning: namespace 'DESeq' is not available and has been replaced
## by .GlobalEnv when processing object 'pasillaGenes'
```

```
countMat <- as.data.frame( pasillaGenes@assayData$counts )
sampleInfo <- as.data.frame( pasillaGenes@phenoData@data )
ddsMat <- DESeqDataSetFromMatrix( countData = countMat,
                                  colData = sampleInfo,
                                  design = ~condition )
```

Questions: How many genes and experiments in the dataset?

Also, for later annotation, this is drosophila samples, so you to install/load this **org.Dm.eg.db** package. More comprehensively, read this document (slightly different approach and followed with earlier version of DESeq analysis, which used the pasilla dataset)
https://bioconductor.org/packages/release/bioc/vignettes/DESeq/inst/doc/DESeq.pdf

---

Example 2b: Polish your report.

The file we print out looks like this way:
baseMean log2FoldChange lfcSE stat pvalue padj
ENSG00000003402 2546.614197 -1.190432416 0.120621217 -9.86917925 5.66E-23 4.82E-21
ENSG00000003987 25.50431871 -1.004804853 0.37265646 -2.696330165 0.007010814 0.034570829
ENSG00000004799 914.3790141 -2.644927284 0.628564617 -4.207884462 2.57773E-05 0.000263293
ENSG00000005471 33.66398824 1.120053429 0.357469028 3.133288034 0.001728597 0.010628632
ENSG00000006283 62.96025142 1.46690803 0.286728811 5.11601197 3.12063E-07 4.68433E-06

Notice that, 1. Column header is missaligned, 2. gene symbols & descriptions are not there 3. Where is the normalized data, and 4. I heard variance stablization normalization, how can I get that?

```
# load three additional libraries, 1) human gene annotation,
#   2) additional functions, 3) easily join data table.
suppressMessages( library(org.Hs.eg.db) )
suppressMessages( library( AnnotationDbi ) )
suppressMessages( library( dplyr ) )

#Check the column of human annotation library column names (ensembl ID?)
columns(org.Hs.eg.db)
```

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

```
anno <- AnnotationDbi::select(org.Hs.eg.db, rownames( sigRes ),
                              columns=c("ENSEMBL", "ENTREZID", "SYMBOL", "GENENAME"),
                              keytype="ENSEMBL")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
# adding ENSEMBL gene ID as a column in significant differentially expression gene table.
sigRes = cbind( ENSEMBL = rownames( sigRes), sigRes )

# left-join, however, make sure the sigRes to be in the data.frame format.
outTable <- left_join( as.data.frame( sigRes ), anno )
```

```
## Joining, by = "ENSEMBL"
```

```
# we may just do that for the entire table...
anno <- AnnotationDbi::select(org.Hs.eg.db, rownames( res ),
                              columns=c("ENSEMBL", "ENTREZID", "SYMBOL", "GENENAME" ),
                              keytype="ENSEMBL")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
# adding ENSEMBL gene ID as a column in significant differentially expression gene table.
res = cbind( ENSEMBL = rownames( res), res )
wholeTable <- left_join( as.data.frame( res ), anno )
```

```
## Joining, by = "ENSEMBL"
```

```
# let's take a look...
head( wholeTable )
```

```
##           ENSEMBL    baseMean log2FoldChange     lfcSE       stat       pvalue
## 1 ENSG00000000003 708.6021697     0.38125398 0.1006544  3.7877523 0.0001520163
## 2 ENSG00000000005   0.0000000             NA        NA         NA           NA
## 3 ENSG00000000419 520.2979006    -0.20681260 0.1122186 -1.8429433 0.0653372915
## 4 ENSG00000000457 237.1630368    -0.03792043 0.1434447 -0.2643558 0.7915057416
## 5 ENSG00000000460  57.9326331     0.08816818 0.2871418  0.3070545 0.7588019240
## 6 ENSG00000000938   0.3180984     1.37822703 3.4998728  0.3937935 0.6937335303
##         padj ENTREZID   SYMBOL
## 1 0.00128363     7105   TSPAN6
## 2         NA    64102     TNMD
## 3 0.19654609     8813     DPM1
## 4 0.91145948    57147    SCYL3
## 5 0.89503278    55732 C1orf112
## 6         NA     2268      FGR
##                                                    GENENAME
## 1                                            tetraspanin 6
## 2                                              tenomodulin
## 3 dolichyl-phosphate mannosyltransferase subunit 1, catalytic
## 4                                   SCY1 like pseudokinase 3
## 5                          chromosome 1 open reading frame 112
## 6              FGR proto-oncogene, Src family tyrosine kinase
```

Question: non-unique table entries.

There is a error message before:
select()' returned **1:many** mapping between keys and columns

What is the problem? Any solutions?
hint: think dplyr::uniuqe()

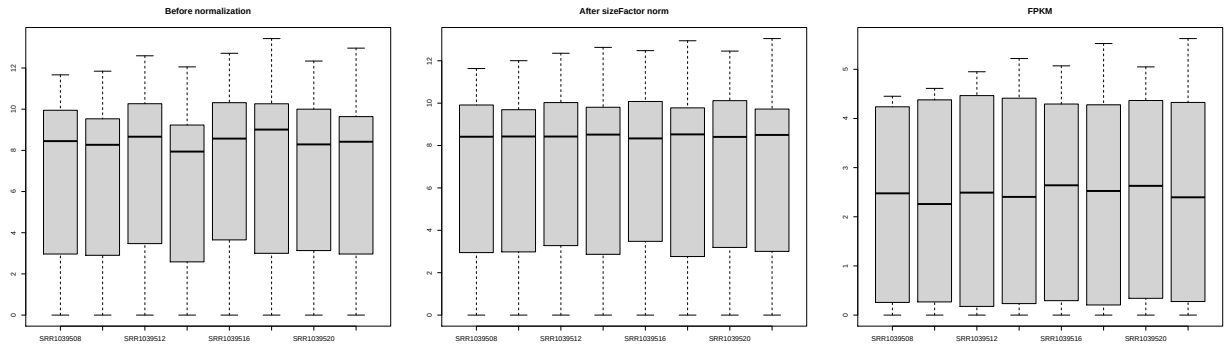
```
# Let's continue the RNA-seq analysis.
# Write again to see if you like this format
write.table( outTable, file = "airway_sigGenes_adjp0.05_lfc1_bm20.txt",
             sep = "\t", col.names = TRUE, row.names = FALSE, quote = FALSE)
write.table( wholeTable, file = "airway_allGenes.txt",
             sep = "\t", col.names = TRUE, row.names = FALSE, quote = FALSE)
```

Example 2c: Additional processed data and plots.

This section only shows some basic functions, you are highly encouraged to join later workshop to explore additional visualization and analysis methods. Our aim is only at obtaining normalized data and common volcano plot.

```
# Some additional steps typically for other purpose (visualization, functional
# analysis, etc). This step has to be done after DESeq() function, or you have
# to call estimateSizeFactors() first.
normData <- counts(dds, normalized=TRUE)

# get gene expression level in fpkm unit.
fpkm <- fpkm( dds, robust=TRUE)

# remove some genes with no value or low expression across all samples.
mu <- apply(normData, 2, mean)
idx <- which( mu > 10 )
```

```
# plot the figure to show the difference before/after normalization.
par( mfrow=c(1,3) )
boxplot( log2( assay(dds[idx,])+1 ), main = "Before normalization" )
boxplot( log2( normData[idx,]+1 ), main = "After sizeFactor norm" )
boxplot( log2( fpkm[idx,]+1 ), main = "FPKM" )
```



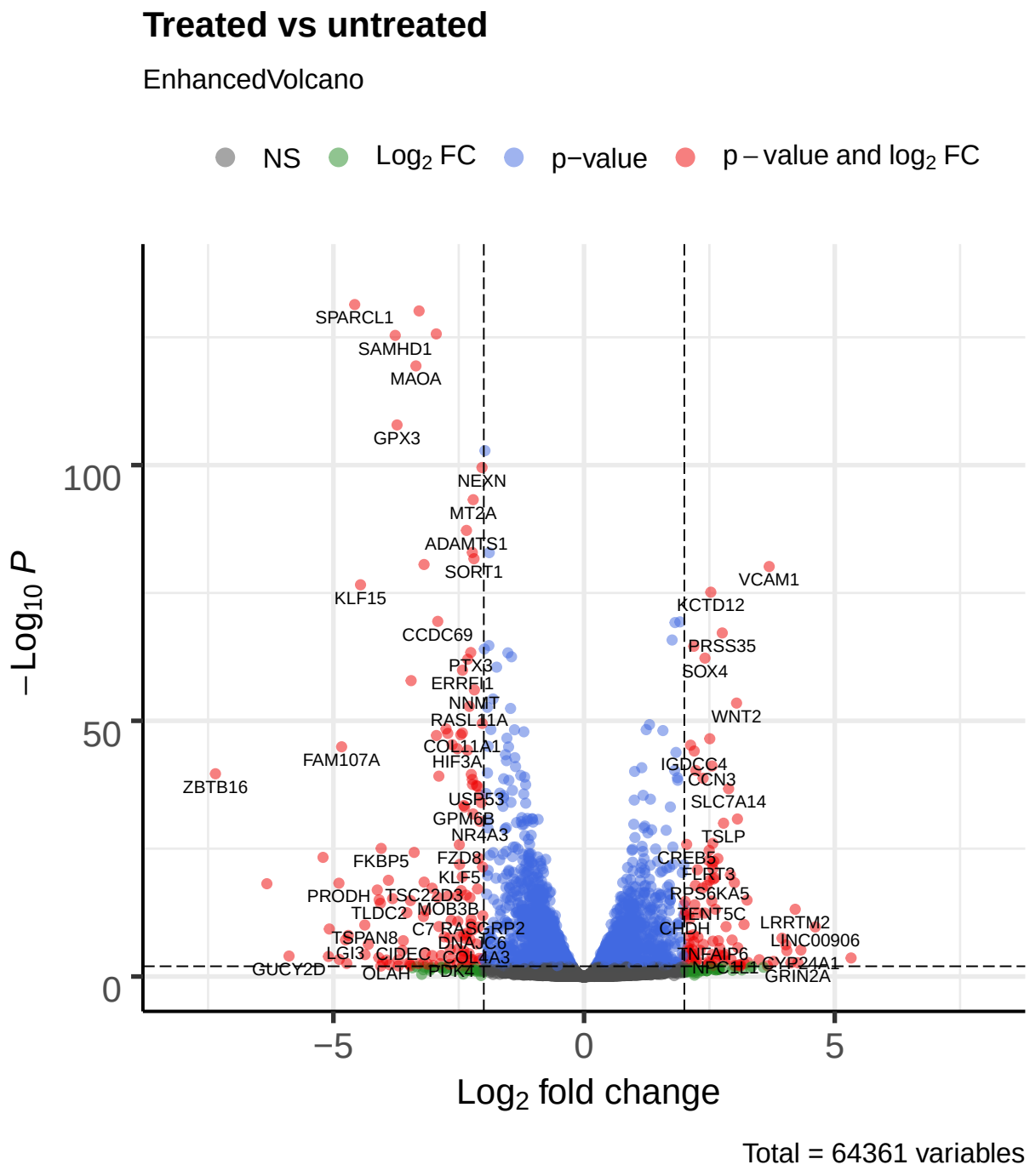

```
# DESeq also introduce an interesting concept: variance stablization transfromation.
# You are more than welcome to join "Visualization and Analysis of Genomic Data" course, where
# we will introduce more normalization methods and reasons behind them.
rld <- rlog( dds )
head( assay( rld ) )
```

```
##                 SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516
## ENSG00000000003   9.399155   9.142506   9.501691   9.320807   9.757189
```

```
## ENSG00000000005   0.000000   0.000000   0.000000   0.000000   0.000000
## ENSG00000000419   8.901297   9.113966   9.032566   9.063920   8.981935
## ENSG00000000457   7.949882   7.882372   7.834287   7.916451   7.773848
## ENSG00000000460   5.849509   5.882324   5.487269   5.770392   5.940321
## ENSG00000000938  -1.369632  -1.367411  -1.305192  -1.362554  -1.338570
##                 SRR1039517 SRR1039520 SRR1039521
## ENSG00000000003   9.512179   9.617365   9.315321
## ENSG00000000005   0.000000   0.000000   0.000000
## ENSG00000000419   9.108522   8.894845   9.052299
## ENSG00000000457   7.886645   7.946396   7.908333
## ENSG00000000460   5.664001   6.107519   5.907764
## ENSG00000000938  -1.373842  -1.367863  -1.368288
```

```
par( mfrow = c( 1, 2 ) )
dds <- estimateSizeFactors(dds)
plot( log2( 1 + normData[ , 1:2] ), col=rgb(0,0,0,.2), pch=16, cex=0.3, main = "log2 transform")
plot( assay(rld)[ , 1:2],           col=rgb(0,0,0,.2), pch=16, cex=0.3, main = "variance stablization" )
```


```
# most people will ask for a volcano plot, here is a pretty version of it.
library( EnhancedVolcano )
EnhancedVolcano( as.data.frame(wholeTable), lab = wholeTable$SYMBOL,
                 x = 'log2FoldChange', y = 'padj',
                 xlim = c(-8, 8), title = 'Treated vs untreated',
                 pCutoff = 0.01, FCcutoff = 2, pointSize = 2.0,
                 labSize = 3.0 )
```

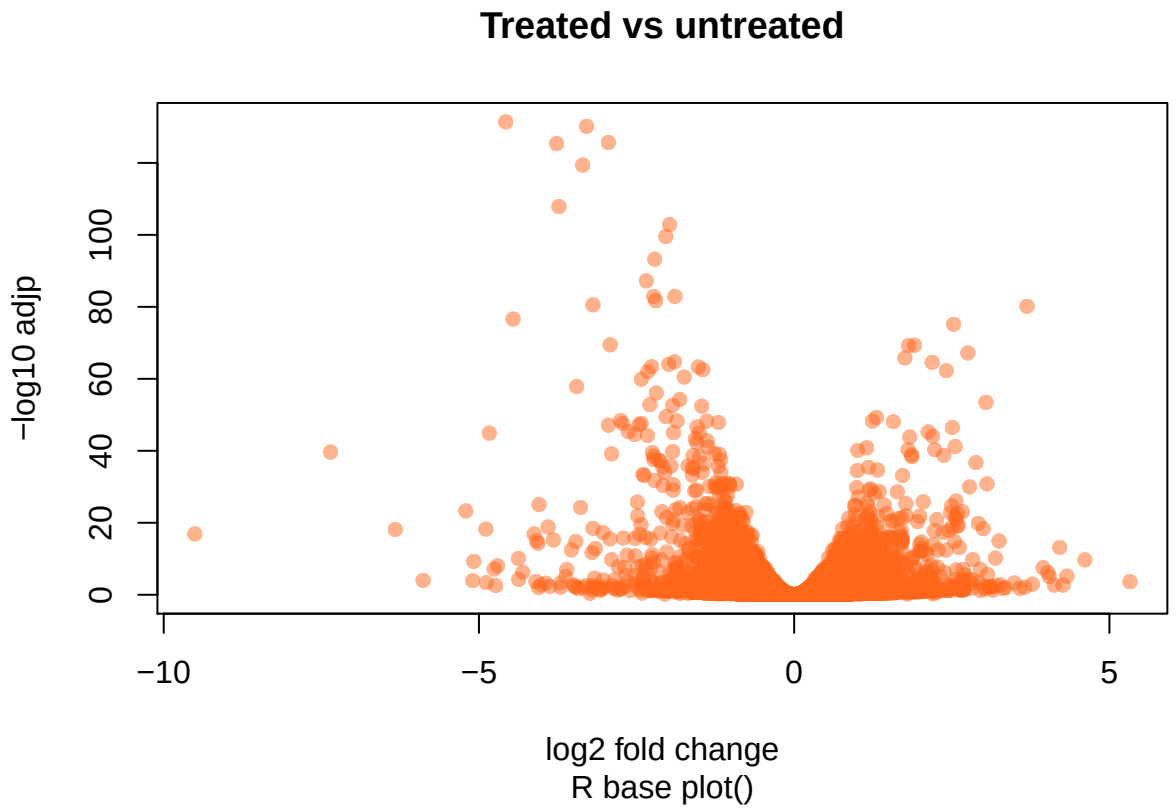


Break #2: How many way you can generate volcalno plots?

We some simple volcano plot template for you to try

```
# Using R base graph.
plot( wholeTable$log2FoldChange, -log10( wholeTable$padj ), pch = 19, cex = 1,
      xlab = "log2 fold change",  ylab = "-log10 adjp", lwd = 0,
      col = rgb(red = 1, green  = 0.4, blue = 0.1,  alpha = 0.5),
```

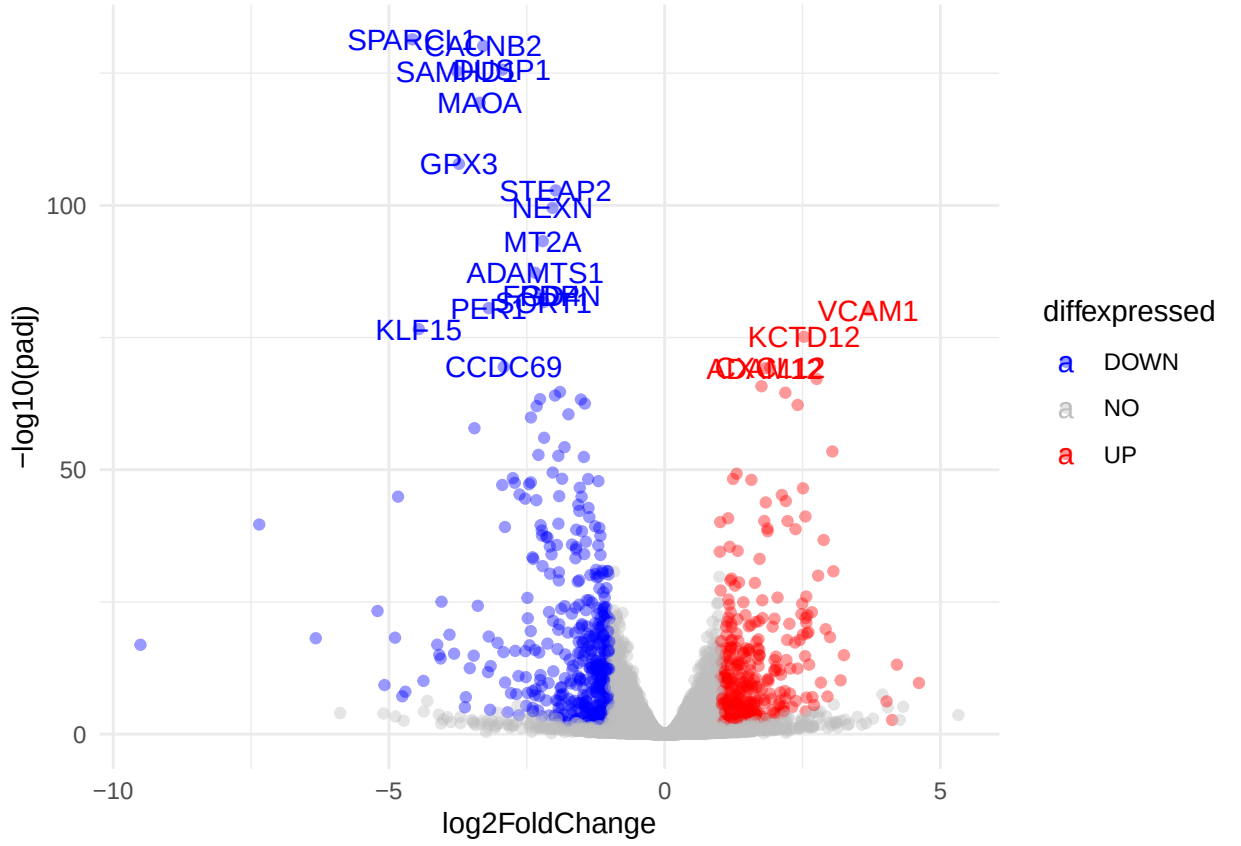
```
      main = "Treated vs untreated", sub = "R base plot()")
```




```
# use ggplot (and dplyr)
top20pThr <- wholeTable$padj[order(wholeTable$padj)[20]]
wt <- wholeTable %>%
  mutate( diffexpressed =
            ifelse(padj <= pThr & baseMean >= baseMeanThr & log2FoldChange >= logFCThr, "UP",
            ifelse(padj <= pThr & baseMean >= baseMeanThr & log2FoldChange <= -logFCThr, "DOWN", "NO" )
  mutate( geneLabel = ifelse( padj <= top20pThr, SYMBOL, NA) )
p <- ggplot( data=wt, aes(x = log2FoldChange, y=-log10(padj),
                     col = diffexpressed, label=geneLabel ) ) +
            geom_point(alpha = 0.4 ) + theme_minimal() + geom_text() +
            scale_color_manual(values=c("blue", "gray", "red"))
p
```

```
## Warning: Removed 46222 rows containing missing values (geom_point).
```

```
## Warning: Removed 64341 rows containing missing values (geom_text).
```




```
# interactive graph using plotly (through ggplot)
# suppressMessages( library(plotly) )
# fig <- ggplotly(p)
# fig
#
# idx = order(wt$padj)[1:5000]
# fig <- plot_ly(wt[idx,], x = ~log2FoldChange, y = ~log2(1+baseMean), z = ~-log10(padj),
#                color = ~log2(1+baseMean), size = ~log2(1+baseMean),
#                sizes = c(5,200), colors = c('#0000FF', '#FF0000') )
# fig <- fig %>% add_markers()
# fig <- fig %>% layout(scene = list(xaxis = list(title = 'logFC'),
#                                    yaxis = list(title = 'baseMean'),
#                                    zaxis = list(title = '-log10 padj')))
# fig
```

Example 3: Using airway data & edgeR

Using edgeR for differential gene expression analysis.

```
# Forth example: airway & edgeR.
suppressMessages( library(edgeR) )

# first set up the DGE list
dge          <- DGEList(counts = assay(airway, "counts"), group = airway$dex)
dge$samples  <- merge(dge$samples, as.data.frame(colData(airway)), by = 0)
```

```
dge$genes    <- data.frame(name = names(rowRanges(airway)), stringsAsFactors = FALSE)

# calcuate size factor normalization. It is mostly the same as DESeq sizeFactor normalzation.
dge <- calcNormFactors(dge)

# Setup the design matrix and estimate the dispersion (variance).
# There are multiple ways to do this, and the weird two-step procedure is necessary.
design <- model.matrix( ~ dge$samples$cell + dge$samples$dex )
design
```

```
##   (Intercept) dge$samples$cellN061011 dge$samples$cellN080611
## 1           1                       0                       0
## 2           1                       0                       0
## 3           1                       0                       0
## 4           1                       0                       0
## 5           1                       0                       1
## 6           1                       0                       1
## 7           1                       1                       0
## 8           1                       1                       0
##   dge$samples$cellN61311 dge$samples$dexuntrt
## 1                      1                    1
## 2                      1                    0
## 3                      0                    1
## 4                      0                    0
## 5                      0                    1
## 6                      0                    0
## 7                      0                    1
## 8                      0                    0
## attr(,"assign")
## [1] 0 1 1 1 2
## attr(,"contrasts")
## attr(,"contrasts")$`dge$samples$cell`
## [1] "contr.treatment"
##
## attr(,"contrasts")$`dge$samples$dex`
## [1] "contr.treatment"
```

```
# estiamte the disperson.
dge <- estimateDisp(dge, design)

#  we do a glmFit() (generalized linear model), similar to LIMMA
fit <- glmFit(dge, design)

# glmLRT conducts likelihood ratio tests for one or more coefficients in the linear model.
# If coef is used, the null hypothesis is that all the coefficients indicated by coef
# are equal to zero.
lrt <- glmLRT( fit, coef = 5 )

# select differentially expressed gene with adj.pval < 0.01.
sigRes2 <- topTags( lrt, n = Inf )
idx <- which( sigRes2$table$logCPM >= log2(cpmThr) &
              abs( sigRes2$table$logFC)>= logFCThr &
              sigRes2$table$FDR <= pThr )
sigRes2 <- sigRes2[idx,]
```

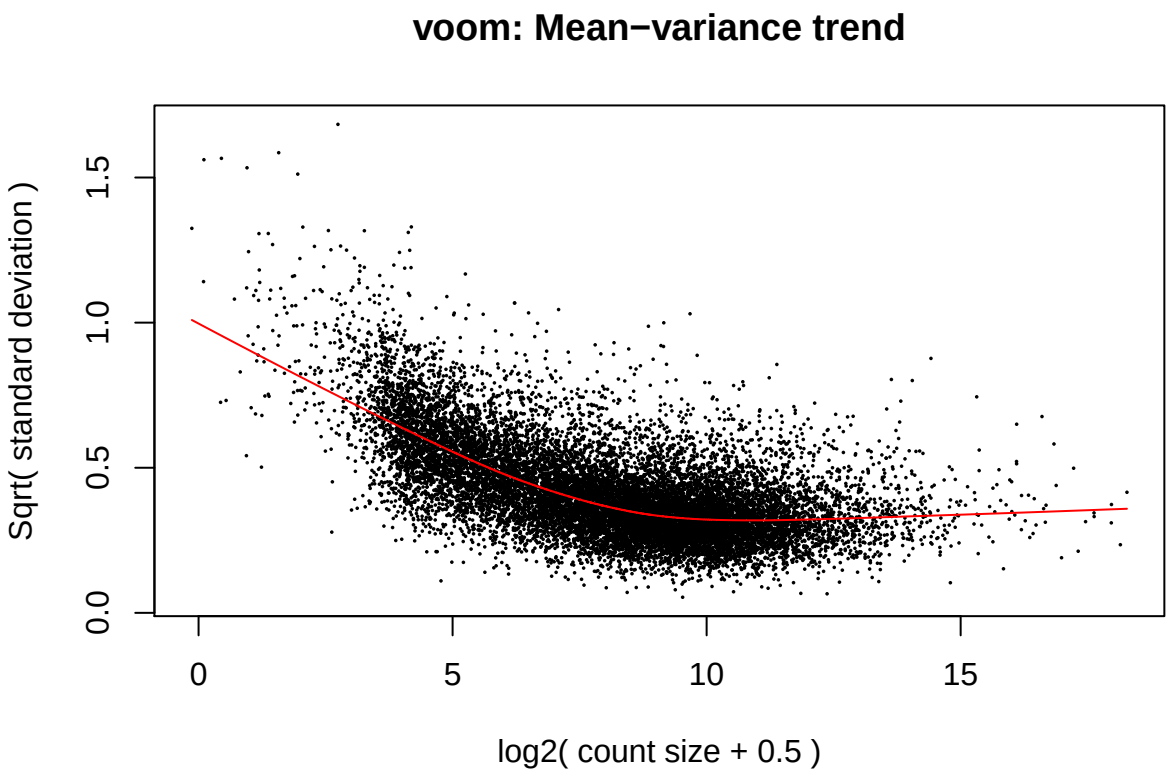
Question:

What is coef = 5 in **glmLRT()** function? Why we don't specify this in **DESeq()** function?

Example 4: Using LIMMA for RNA-seq analysis.

voom is an acronym for mean-variance modelling at the observational level. Raw counts show increasing variance with increasing count size, while log-counts typically show a decreasing mean-variance trend. Voom() function estimates the mean-variance trend for log-counts, then assigns a weight to each observation based on its predicted variance. The weights are then used in the linear modelling process to adjust for heteroscedasticity.

```
# Using counts per million (CPM) to filter.
library( limma )
drop <- which(apply(cpm(dge), 1, max) <= cpmThr )
dge_voom <- dge[-drop,]

# apply voom transfromation. You do need to look at the curve the
# make sure the regression is smooth, otherwise, you need to drop (trim)
# your data more aggressive (increase cpmThr in this case)
par( mfrow=c(1,1) )  # reset to 1 figure per page.

# Voom recommend filtering more aggressively before sizefactor normalization.
y <- voom( dge_voom, design, plot = T)
```

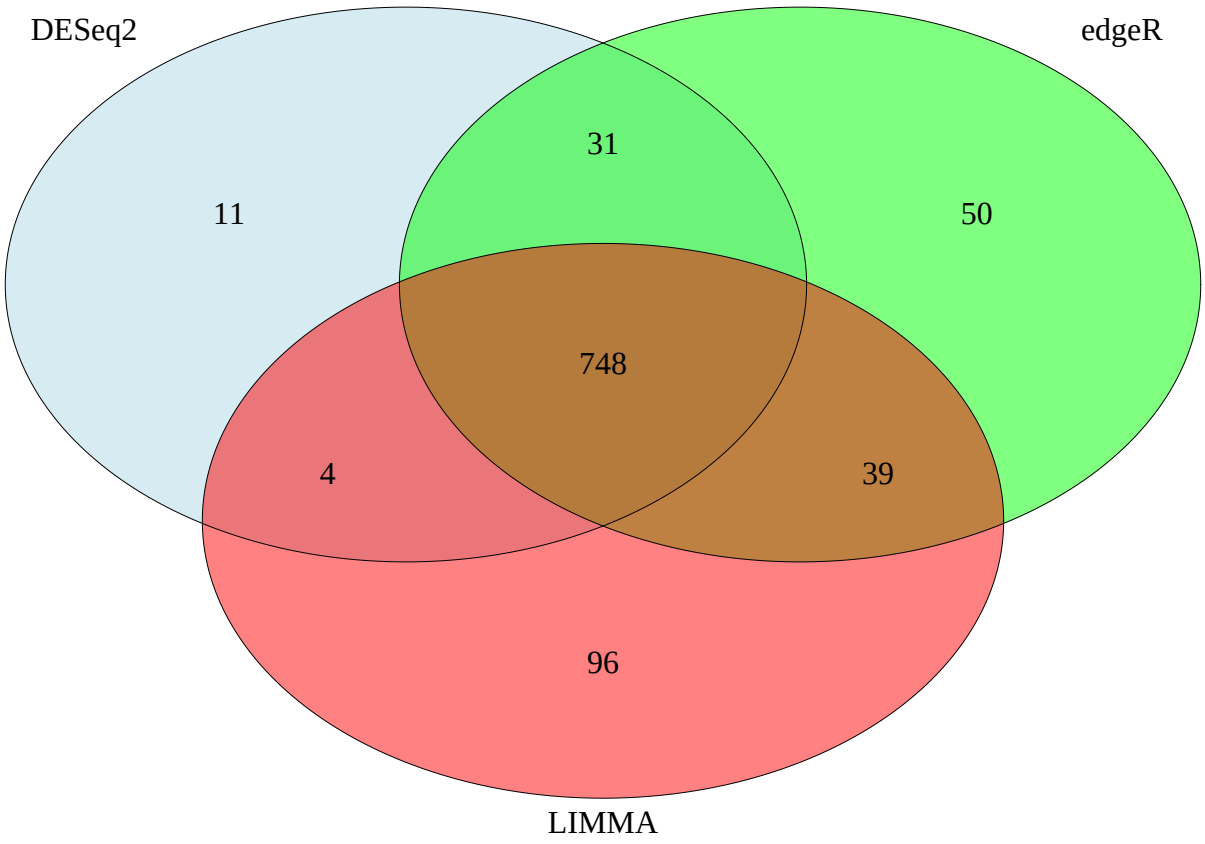



```
# linear model, and empircal Bayes estiamte.
fit <- lmFit( y, design )
fit <- eBayes( fit )

# select differentially expressed genes.
sigRes3 <- topTable(fit, coef=ncol(design), p.value = pThr, n = Inf )
idx <- which( abs( sigRes3$logFC ) >= logFCThr )
sigRes3 <- sigRes3[idx, ]
```

Summary

Differentially expressed genes from 3 algorithms: are they similar? To evaluate them, you can use Venn Diagram to compare all genes.

```
# overlapping between 3 algorithms.
suppressMessages( library(VennDiagram) )
suppressMessages( library(grid) )
gg <- venn.diagram( x = list( sigRes$ENSEMBL, sigRes2$table$name, sigRes3$name ),
              category.names = c("DESeq2" , "edgeR", "LIMMA"), lwd = 0,
              fill = c("lightblue", "green", "red"),
              filename = NULL,
              height = 1000, width = 1000 )
grid.draw(gg)
```



Again, wrapping up this hands-on session, we went through,

1. One interactive tool for microarray data analysis to capture the essence of gene expression data analysis, which is

- Transformation (log2)
- Normalization (quantile, sizeFactor, median, etc)
- Differential expression (DESeq, edge, LIMMA, etc)
- Visualization

2. Examples how to use DESeq2 to perform differential gene expression analysis
3. Gene annotation using R annotation packages
4. Report of your analysis results and visualization (a lot of variation of volcano plots)
5. Other RNA-seq analysis packages (edgeR, LIMMA-Voom)

We **did not** cover

1. Mathematical background of RNA-seq (Other than "Read Count" follow non-negative binomial distribution)

2. We did not cover how to create design matrix (please consult biostatisticians)

3. We did not provide common **visualization** methods, such as PCA, MDS, hierarchical clustering, or more modern methods such as tSNE (Pleasse join **Workshop 7**)

4. We did not cover functional analysis (please join **Workshop 8**)

5. Public-domain RNA-seq data and methods to access them (**Workshop 9**)
6. Sequence alignment will also be covered in later workshops (**Workshop 11**)

7. We certainly did not discuss important R data classes (DESeqDataSet, DEGList, and others). We will provide R language class in 2021 academic year.

See you all in the next lecture. Also look for "Visualization and Analysis of Genomic Data" course.

NCBI GEO datasets: Directly download gene expression data from GEO.

You can also use command line function (after install package "Phantasus") to extract gene expression data from NCBI GEO database. RNA-seq, however, does not work with method (other than metadata section) since NBCI GEO does not offer consistent gene expression level matrix (majority of them only have metadata). **Lecture 9** will provide alternative methods for RNA-seq. There are many packages for this purpose, such as "GEOquery" package.

```
# Directly download very first GSE dataset from GEO database
gse1 <- getGSE( "GSE1" )[[1]]
```

```
## Trying www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE1&targ=self&form=text&view=brief
```

```
## No encoding supplied: defaulting to UTF-8.
```

```
## Stored brief data of GSE1 at C:/Users/mille/AppData/Local/Temp/RtmpqI4261/geo/series/GSEnnn/GSE1/bri
```

```
##
## -- Column specification --------------------------------------------------------------------------
## cols(
##   .default = col_double()
## )
## i Use `spec()` for the full column specifications.
```

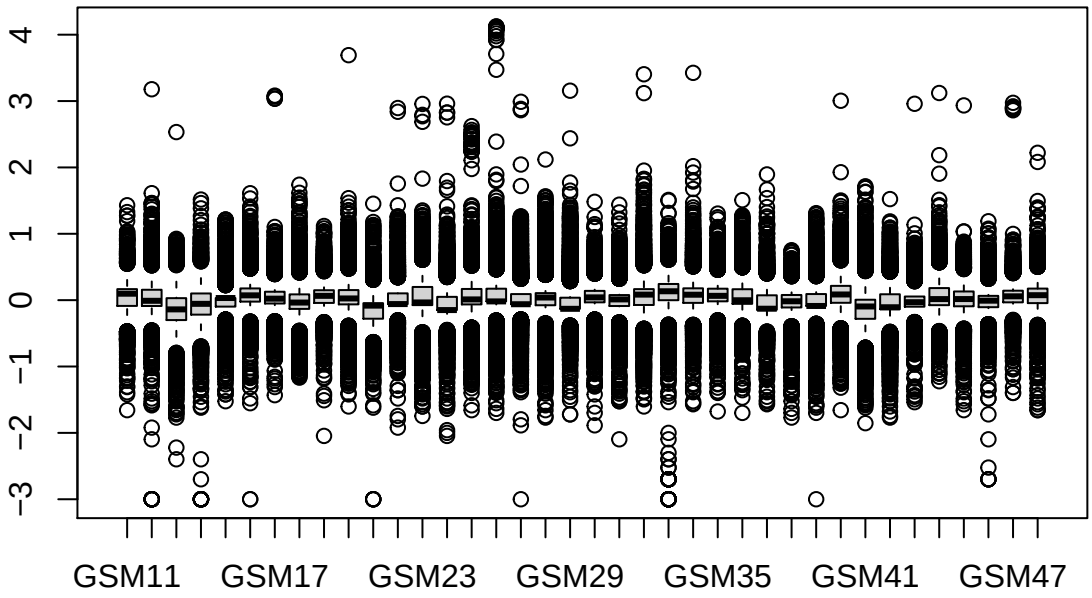
```
exprs <- gse1@assayData$exprs

# we can look at the abstract of this dataset
gse1@experimentData@abstract
```

```
## [1] "This series represents a group of cutaneous malignant melanomas and unrelated controls which we
```

```
gse1@experimentData@pubMedIds
```

```
## [1] "10952317"
```

```
# boxplot of the entire 38 samples.
boxplot( exprs )
```




```
# Session Info.
sessionInfo()
```

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18363)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
##  [1] grid      stats4    parallel  stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] VennDiagram_1.6.20          futile.logger_1.4.3
##  [3] edgeR_3.30.3                limma_3.44.3
##  [5] EnhancedVolcano_1.6.0       ggrepel_0.8.2
##  [7] dplyr_1.0.2                 org.Hs.eg.db_3.11.4
##  [9] AnnotationDbi_1.50.3        pasilla_1.16.0
## [11] ggplot2_3.3.2               DESeq2_1.28.1
```

```
## [13] airway_1.8.0                SummarizedExperiment_1.18.2
## [15] DelayedArray_0.14.1         matrixStats_0.57.0
## [17] Biobase_2.48.0              GenomicRanges_1.40.0
## [19] GenomeInfoDb_1.24.2         IRanges_2.22.2
## [21] S4Vectors_0.26.1            BiocGenerics_0.34.0
## [23] phantasus_1.8.0
##
## loaded via a namespace (and not attached):
##  [1] fgsea_1.14.0           colorspace_1.4-1       ellipsis_0.3.1
##  [4] XVector_0.28.0         rstudioapi_0.11        farver_2.0.3
##  [7] ccaPP_0.3.3            bit64_4.0.5            fansi_0.4.1
## [10] mvtnorm_1.1-1          xml2_1.3.2             codetools_0.2-16
## [13] splines_4.0.2          robustbase_0.93-6      geneplotter_1.66.0
## [16] knitr_1.30             jsonlite_1.7.1         annotate_1.66.0
## [19] httr_1.4.2             readr_1.4.0            compiler_4.0.2
## [22] assertthat_0.2.1       Matrix_1.2-18          cli_2.0.2
## [25] later_1.1.0.1          formatR_1.7            htmltools_0.5.0
## [28] tools_4.0.2            gtable_0.3.0           glue_1.4.2
## [31] GenomeInfoDbData_1.2.3 rappdirs_0.3.1         fastmatch_1.1-0
## [34] Rcpp_1.0.5             vctrs_0.3.4            svglite_1.2.3.2
## [37] xfun_0.18              stringr_1.4.0          webutils_1.1
## [40] mime_0.9               lifecycle_0.2.0        XML_3.99-0.5
## [43] DEoptimR_1.0-8         zlibbioc_1.34.0        scales_1.1.1
## [46] hms_0.5.3              promises_1.1.1         rhdf5_2.32.4
## [49] GEOquery_2.56.0        lambda.r_1.2.4         RColorBrewer_1.1-2
## [52] curl_4.3               yaml_2.2.1             memoise_1.1.0
## [55] gridExtra_2.3          gdtools_0.2.2          stringi_1.5.3
## [58] RSQLite_2.2.0          genefilter_1.70.0      Rook_1.1-1
## [61] pcaPP_1.9-73           opencpu_2.2.0          BiocParallel_1.22.0
## [64] rlang_0.4.7            pkgconfig_2.0.3        systemfonts_0.3.2
## [67] bitops_1.0-6           evaluate_0.14          lattice_0.20-41
## [70] purrr_0.3.4            Rhdf5lib_1.10.1        labeling_0.3
## [73] bit_4.0.4              tidyselect_1.1.0       magrittr_1.5
## [76] R6_2.4.1               generics_0.0.2         DBI_1.1.0
## [79] pillar_1.4.6           withr_2.3.0            survival_3.1-12
## [82] RCurl_1.98-1.2         tibble_3.0.3           crayon_1.3.4
## [85] futile.options_1.0.1   rmarkdown_2.4          locfit_1.5-9.4
## [88] data.table_1.13.0      blob_1.2.1             digest_0.6.25
## [91] xtable_1.8-4           tidyr_1.1.2            httpuv_1.5.4
## [94] brew_1.0-6             openssl_1.4.3          munsell_0.5.0
## [97] askpass_1.1
```