Differential Expression Analysis with EdgeR

Carson Stacy & Jeffrey Lewis

Fall 2023

last updated: 2023-09-19

Getting Things Setup

As usual, make sure we have the right packages for this exercise

```
if (!require("pacman")) install.packages("pacman"); library(pacman)
```

```
## Loading required package: pacman
```

```
# let's load all of the files we were using and want to have again today
p_load("tidyverse", "knitr", "readr",
       "pander", "BiocManager",
       "dplyr", "stringr",
       "statmod", # required dependency, need to load manually on some macOS versions.
       "purrr", # for working with lists (beautify column names)
       "webshot2", # allow for pdf of output table.
       "reactable") # for pretty tables.

# We also need these Bioconductor packages today.
p_load("edgeR", "AnnotationDbi", "org.Sc.sgd.db")
```

Description

This will be our first differential expression analysis workflow, converting gene counts across samples into meaningful information about genes that appear to be significantly differentially expressed between samples

Learning outcomes

At the end of this exercise, you should be able to:

- Generate a table of sample metadata.
- Filter low counts and normalize count data.
- Utilize the edgeR package to identify differentially expressed genes.

```
library(edgeR)
library(org.Sc.sgd.db)
# for ease of use, set max number of digits after decimal
options(digits=3)
```

Loading in the featureCounts object

We saved this file in the last exercise (0Read_Counting.Rmd) from the RSubread package. Now we can load that object back in and assign it to the variable `fc`. Be sure to change the file path if you have saved it in a different location.

```
path_fc_object <- path.expand("~/Desktop/Genomic_Data_Analysis/Data/Counts/Rsubread/rsubread.yeast_fc_ou

counts_subset <- readRDS(file = path_fc_object)$counts
```

We generated those counts on a subset of the fastq files, but we can load the complete count file with the command below. This file has been generated with the full size fastq files with Salmon.

```
counts <- read_tsv('https://github.com/clstacy/GenomicDataAnalysis_Fa23/raw/main/data/ethanol_stress/cou
                   col_names = TRUE) %>%
  # when we saved the tsv file, it converted the rownames to a column,
  # we are converting it back with this piped command.
  column_to_rownames("Name")
```

```
## Rows: 6571 Columns: 17
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (1): Name
## dbl (16): YPS606_MSN24_ETOH_REP1_R1.fastq.gz_quant, YPS606_MSN24_ETOH_REP2_R...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

So far, we've been able to process all of the fastq files without much information about what each sample is in the experimental design. Now, we need the metadata for the samples. Note that the order matters for these files

To find the order of files we need, we can get just the part of the column name before the first "." symbol with this command:

```
str_split_fixed(counts %>% colnames(), "\\.", n = 2)[, 1] %>% cat()
```

```
## YPS606_MSN24_ETOH_REP1_R1 YPS606_MSN24_ETOH_REP2_R1 YPS606_MSN24_ETOH_REP3_R1 YPS606_MSN24_ETOH_REP4_
```

```
sample_metadata <- tribble(
  ~Sample,                      ~Genotype,    ~Condition,
  "YPS606_MSN24_ETOH_REP1_R1",   "msn24dd",   "EtOH",
  "YPS606_MSN24_ETOH_REP2_R1",   "msn24dd",   "EtOH",
  "YPS606_MSN24_ETOH_REP3_R1",   "msn24dd",   "EtOH",
  "YPS606_MSN24_ETOH_REP4_R1",   "msn24dd",   "EtOH",
```

```
  "YPS606_MSN24_MOCK_REP1_R1",   "msn24dd",   "unstressed",
  "YPS606_MSN24_MOCK_REP2_R1",   "msn24dd",   "unstressed",
  "YPS606_MSN24_MOCK_REP3_R1",   "msn24dd",   "unstressed",
  "YPS606_MSN24_MOCK_REP4_R1",   "msn24dd",   "unstressed",
  "YPS606_WT_ETOH_REP1_R1",      "WT",        "EtOH",
  "YPS606_WT_ETOH_REP2_R1",      "WT",        "EtOH",
  "YPS606_WT_ETOH_REP3_R1",      "WT",        "EtOH",
  "YPS606_WT_ETOH_REP4_R1",      "WT",        "EtOH",
  "YPS606_WT_MOCK_REP1_R1",      "WT",        "unstressed",
  "YPS606_WT_MOCK_REP2_R1",      "WT",        "unstressed",
  "YPS606_WT_MOCK_REP3_R1",      "WT",        "unstressed",
  "YPS606_WT_MOCK_REP4_R1",      "WT",        "unstressed") %>%
  # Create a new column that combines the Genotype and Condition value
  mutate(Group = factor(
    paste(Genotype, Condition, sep = "."),
    levels = c(
      "WT.unstressed","WT.EtOH",
      "msn24dd.unstressed", "msn24dd.EtOH"
    )
  )) %>%
  # make Condition and Genotype a factor (with baseline as first level) for edgeR
  mutate(
    Genotype = factor(Genotype,
                      levels = c("WT", "msn24dd")),
    Condition = factor(Condition,
                       levels = c("unstressed", "EtOH"))
  )
```

Now, let's create a design matrix with this information

```
group <- sample_metadata$Group
design <- model.matrix(~ 0 + group)
design
```

```
##    groupWT.unstressed groupWT.EtOH groupmsn24dd.unstressed groupmsn24dd.EtOH
## 1                   0           0                       0                 1
## 2                   0           0                       0                 1
## 3                   0           0                       0                 1
## 4                   0           0                       0                 1
## 5                   0           0                       1                 0
## 6                   0           0                       1                 0
## 7                   0           0                       1                 0
## 8                   0           0                       1                 0
## 9                   0           1                       0                 0
## 10                  0           1                       0                 0
## 11                  0           1                       0                 0
## 12                  0           1                       0                 0
## 13                  1           0                       0                 0
## 14                  1           0                       0                 0
## 15                  1           0                       0                 0
## 16                  1           0                       0                 0
## attr(,"assign")
## [1] 1 1 1 1
```

```
## attr(,"contrasts")
## attr(,"contrasts")$group
## [1] "contr.treatment"
```

```
colnames(design) <- levels(group)
design
```

```
##    WT.unstressed WT.EtOH msn24dd.unstressed msn24dd.EtOH
## 1              0       0                  0           1
## 2              0       0                  0           1
## 3              0       0                  0           1
## 4              0       0                  0           1
## 5              0       0                  1           0
## 6              0       0                  1           0
## 7              0       0                  1           0
## 8              0       0                  1           0
## 9              0       1                  0           0
## 10             0       1                  0           0
## 11             0       1                  0           0
## 12             0       1                  0           0
## 13             1       0                  0           0
## 14             1       0                  0           0
## 15             1       0                  0           0
## 16             1       0                  0           0
## attr(,"assign")
## [1] 1 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$group
## [1] "contr.treatment"
```

Count loading and Annotation

The count matrix is used to construct a DGEList class object. This is the main data class in the edgeR package. The DGEList object is used to store all the information required to fit a generalized linear model to the data, including library sizes and dispersion estimates as well as counts for each gene.

```
y <- DGEList(counts, group=group)
colnames(y) <- sample_metadata$Sample
y$samples
```

```
##                                      group lib.size norm.factors
## YPS606_MSN24_ETOH_REP1_R1       msn24dd.EtOH 17409481            1
## YPS606_MSN24_ETOH_REP2_R1       msn24dd.EtOH 14055425            1
## YPS606_MSN24_ETOH_REP3_R1       msn24dd.EtOH 13127876            1
## YPS606_MSN24_ETOH_REP4_R1       msn24dd.EtOH 16655559            1
## YPS606_MSN24_MOCK_REP1_R1 msn24dd.unstressed 12266723            1
## YPS606_MSN24_MOCK_REP2_R1 msn24dd.unstressed 11781244            1
## YPS606_MSN24_MOCK_REP3_R1 msn24dd.unstressed 11340274            1
## YPS606_MSN24_MOCK_REP4_R1 msn24dd.unstressed 13024330            1
## YPS606_WT_ETOH_REP1_R1               WT.EtOH 15422048            1
## YPS606_WT_ETOH_REP2_R1               WT.EtOH 14924728            1
```

```
## YPS606_WT_ETOH_REP3_R1              WT.EtOH 14738753            1
## YPS606_WT_ETOH_REP4_R1              WT.EtOH 12203133            1
## YPS606_WT_MOCK_REP1_R1        WT.unstressed 13592206            1
## YPS606_WT_MOCK_REP2_R1        WT.unstressed 12921965            1
## YPS606_WT_MOCK_REP3_R1        WT.unstressed 13128396            1
## YPS606_WT_MOCK_REP4_R1        WT.unstressed 15568155            1
```

Human-readable gene symbols can also be added to complement the gene ID for each gene, using the annotation in the org.Sc.sgd.db package.

```
y$genes <- AnnotationDbi::select(org.Sc.sgd.db,keys=rownames(y),columns="GENENAME")
```

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

```
head(y$genes)
```

```
##       ORF        SGD GENENAME
## 1 YIL170W S000001432    HXT12
## 2 YIL175W S000001437     <NA>
## 3 YPL276W S000006197     <NA>
## 4 YFL056C S000001838     AAD6
## 5 YCL074W S000000579     <NA>
## 6 YAR061W S000000087     <NA>
```

Filtering to remove low counts

Genes with very low counts across all libraries provide little evidence for differential ex- pression. In addition, the pronounced discreteness of these counts interferes with some of the statistical approximations that are used later in the pipeline. These genes should be filtered out prior to further analysis. Here, we will retain a gene only if it is expressed at a count-per-million (CPM) above 0.7 in at least four samples.

```
keep <- rowSums(cpm(y) > 0.7) >= 4
y <- y[keep,]
summary(keep)
```

```
##    Mode   FALSE    TRUE
## logical     956    5615
```

Where did those cutoff numbers come from?

As a general rule, we don't want to exclude a gene that is expressed in only one group, so a cutoff number equal to the number of replicates can be a good starting point. For counts, a good threshold can be chosen by identifying the CPM that corresponds to a count of 10, which in this case would be about 0.7:

```
cpm(10, mean(y$samples$lib.size))
```

```
##      [,1]
## [1,] 0.72
```

Smaller CPM thresholds are usually appropriate for larger libraries.

Normalization for composition bias

TMM normalization is performed to eliminate composition biases between libraries. This generates a set of normalization factors, where the product of these factors and the library sizes defines the effective library size. The calcNormFactors function returns the DGEList argument with only the norm.factors changed.

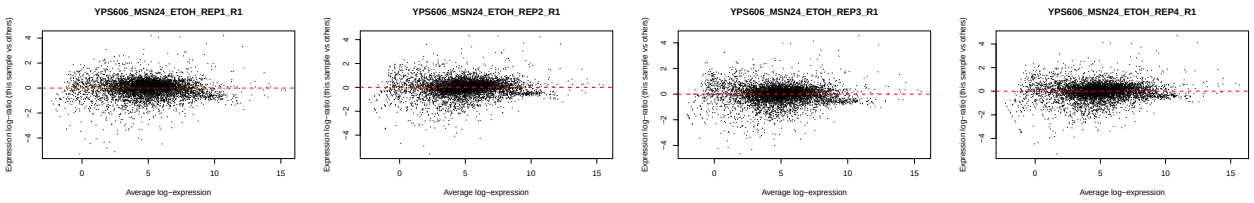
```
y <- calcNormFactors(y)
y$samples
```

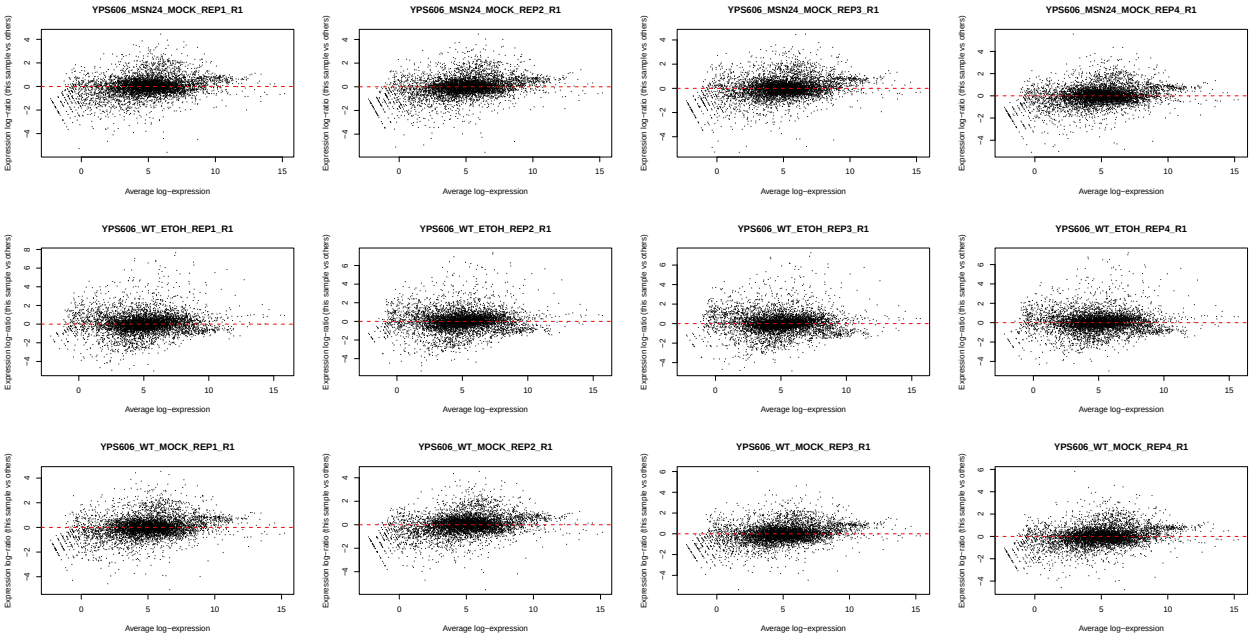
```
##                                      group lib.size norm.factors
## YPS606_MSN24_ETOH_REP1_R1       msn24dd.EtOH 17409481        1.239
## YPS606_MSN24_ETOH_REP2_R1       msn24dd.EtOH 14055425        1.102
## YPS606_MSN24_ETOH_REP3_R1       msn24dd.EtOH 13127876        1.108
## YPS606_MSN24_ETOH_REP4_R1       msn24dd.EtOH 16655559        1.007
## YPS606_MSN24_MOCK_REP1_R1 msn24dd.unstressed 12266723        1.038
## YPS606_MSN24_MOCK_REP2_R1 msn24dd.unstressed 11781244        1.003
## YPS606_MSN24_MOCK_REP3_R1 msn24dd.unstressed 11340274        0.960
## YPS606_MSN24_MOCK_REP4_R1 msn24dd.unstressed 13024330        0.984
## YPS606_WT_ETOH_REP1_R1               WT.EtOH 15422048        0.839
## YPS606_WT_ETOH_REP2_R1               WT.EtOH 14924728        0.941
## YPS606_WT_ETOH_REP3_R1               WT.EtOH 14738753        0.988
## YPS606_WT_ETOH_REP4_R1               WT.EtOH 12203133        0.971
## YPS606_WT_MOCK_REP1_R1         WT.unstressed 13592206        0.990
## YPS606_WT_MOCK_REP2_R1         WT.unstressed 12921965        1.038
## YPS606_WT_MOCK_REP3_R1         WT.unstressed 13128396        0.900
## YPS606_WT_MOCK_REP4_R1         WT.unstressed 15568155        0.951
```

The normalization factors multiply to unity across all libraries. A normalization factor below unity indicates that the library size will be scaled down, as there is more suppression (i.e., composition bias) in that library relative to the other libraries. This is also equivalent to scaling the counts upwards in that sample. Conversely, a factor above unity scales up the library size and is equivalent to downscaling the counts. The performance of the TMM normalization procedure can be examined using mean- difference (MD) plots. This visualizes the library size-adjusted log-fold change between two libraries (the difference) against the average log-expression across those libraries (the mean). The below command plots an MD plot, comparing sample 1 against an artificial library constructed from the average of all other samples.

MDS plots

```
for (sample in 1:nrow(y$samples)) {
  plotMD(cpm(y, log=TRUE), column=sample)
  abline(h=0, col="red", lty=2, lwd=2)
}
```

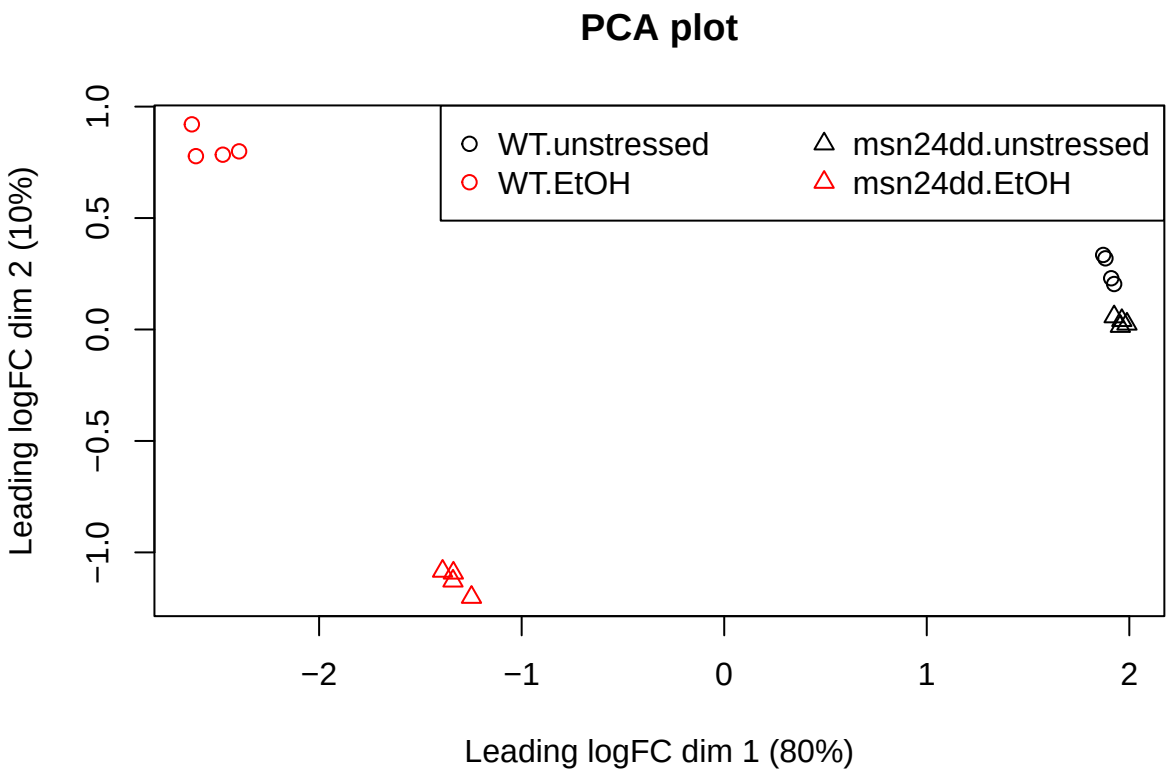




Exploring differences between libraries

The data can be explored by generating multi-dimensional scaling (MDS) plots. This visualizes the differences between the expression profiles of different samples in two dimensions. The next plot shows the MDS plot for the yeast heatshock data.

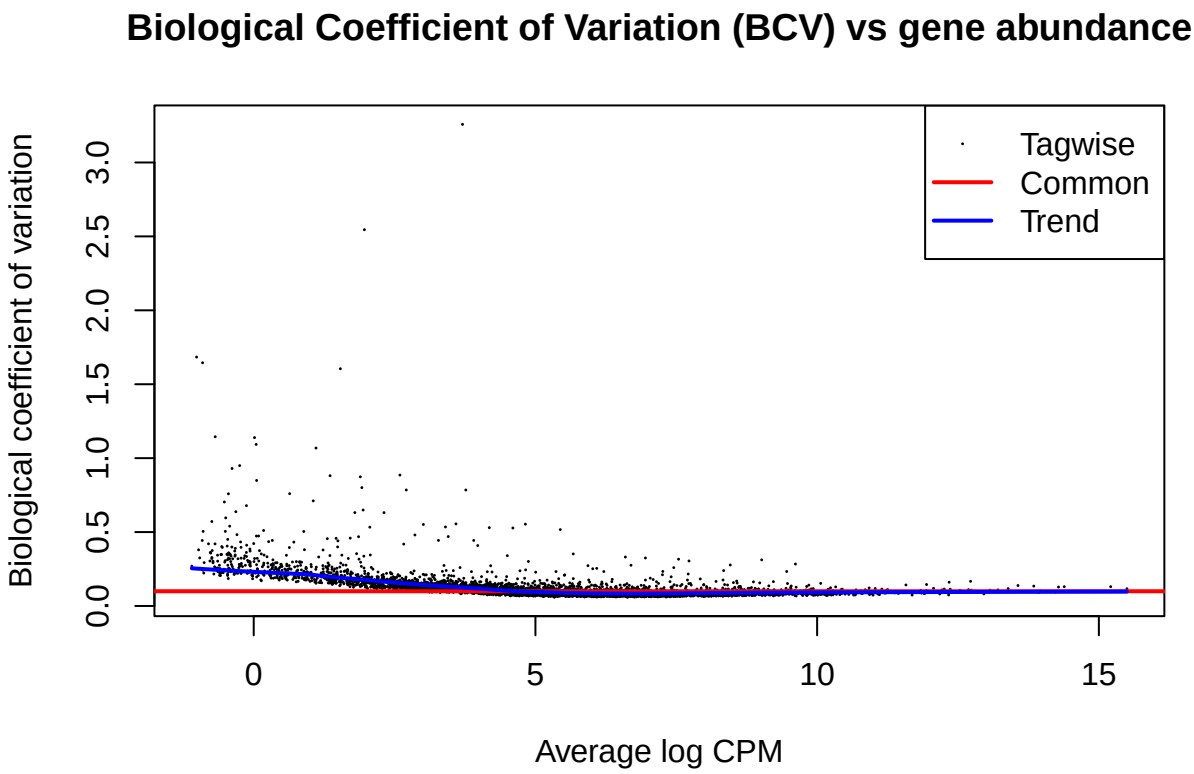
```
points <- c(1,1,2,2)
colors <- rep(c("black", "red"),8)
plotMDS(y, col=colors[group], pch=points[group])
legend("topright", legend=levels(group),
     pch=points, col=colors, ncol=2)
title(main="PCA plot")
```



Estimate Dispersion

The trended NB dispersion is estimated using the estimateDisp function. This returns the DGEList object with additional entries for the estimated NB dispersions for all genes. These estimates can be visualized with plotBCV, which shows the root-estimate, i.e., the biological coefficient of variation for each gene

```
y <- estimateDisp(y, design, robust=TRUE)
plotBCV(y)
title(main="Biological Coefficient of Variation (BCV) vs gene abundance")
```



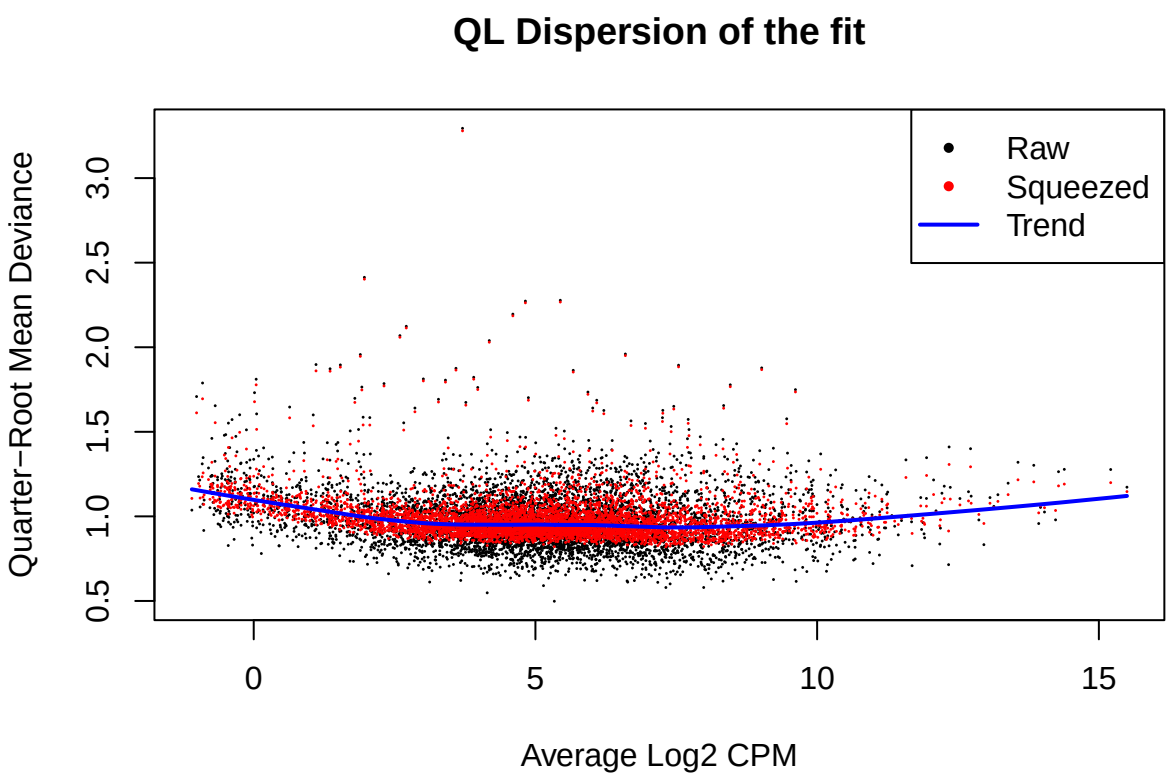
In general, the trend in the NB dispersions should decrease smoothly with increasing abundance. This is because the expression of high-abundance genes is expected to be more stable than that of low-abundance genes. Any substantial increase at high abundances may be indicative of batch effects or trended biases. The value of the trended NB dispersions should range between 0.005 to 0.05 for laboratory-controlled biological systems like mice or cell lines, though larger values will be observed for patient-derived data (> 0.1)

For the QL dispersions, estimation can be performed using the glmQLFit function. This returns a DGEGLM object containing the estimated values of the GLM coefficients for each gene

```
fit <- glmQLFit(y, design, robust=TRUE)
head(fit$coefficients)
```

```
##         WT.unstressed WT.EtOH msn24dd.unstressed msn24dd.EtOH
## YIL170W         -15.1  -13.10             -15.87       -13.21
## YFL056C         -11.1  -11.01             -11.07       -10.42
## YAR061W         -13.7  -13.30             -13.36       -13.36
## YGR014W          -8.5   -8.74              -8.41        -8.66
## YPR031W         -10.5  -11.89             -10.45       -11.86
## YIL003W         -10.6  -12.07             -10.72       -11.97
```

```
plotQLDisp(fit)
title(main="QL Dispersion of the fit")
```



EB squeezing of the raw dispersion estimators towards the trend reduces the uncertainty of the final estimators. The extent of this moderation is determined by the value of the prior df, as estimated from the data. Large estimates for the prior df indicate that the QL dispersions are less variable between genes, meaning that stronger EB moderation can be performed. Small values for the prior df indicate that the dispersions are highly variable, meaning that strong moderation would be inappropriate

Setting `robust=TRUE` in glmQLFit is strongly recommended. This causes glmQLFit to estimate a vector of df.prior values, with lower values for outlier genes and larger values for the main body of genes.

Testing for differential expression

The final step is to actually test for significant differential expression in each gene, using the QL F-test. The contrast of interest can be specified using the makeContrasts function. Here, genes are detected that are DE between the stressed and unstressed. This is done by defining the null hypothesis as heat stressed - unstressed = 0.

```
# generate contrasts we are interested in learning about
my.contrasts <- makeContrasts(EtOHvsMOCK.WT = WT.EtOH - WT.unstressed,
                      EtOHvsMOCK.MSN24dd = msn24dd.EtOH - msn24dd.unstressed,
                      EtOH.MSN24ddvsWT = msn24dd.EtOH - WT.EtOH,
                      MOCK.MSN24ddvsWT = msn24dd.unstressed - WT.unstressed,
                      EtOHvsWT.MSN24ddvsWT = (msn24dd.EtOH-msn24dd.unstressed)-(WT.EtOH-WT.unstressed),
                      levels=design)
```

```r
# This contrast looks at the difference in the stress responses between mutant and WT
res <- glmQLFTest(fit, contrast = my.contrasts[,"EtOHvsWT.MSN24ddvsWT"])
```

```r
# let's take a quick look at the results
topTags(res, n=10)
```

```
## Coefficient:  1*WT.unstressed -1*WT.EtOH -1*msn24dd.unstressed 1*msn24dd.EtOH
##             ORF        SGD GENENAME logFC logCPM    F   PValue      FDR
## YMR105C YMR105C S000004711     PGM2 -6.84   9.70 1608 2.91e-24 1.64e-20
## YMR196W YMR196W S000004809     <NA> -5.15   8.36  877 5.58e-21 1.06e-17
## YKL035W YKL035W S000001518     UGP1 -3.84  10.78  868 5.65e-21 1.06e-17
## YDR516C YDR516C S000002924     EMI2 -4.01   9.08  795 1.65e-20 2.31e-17
## YBR126C YBR126C S000000330     TPS1 -3.46   9.81  693 8.80e-20 9.32e-17
## YLR258W YLR258W S000004248     GSY2 -4.86   8.25  680 1.10e-19 9.32e-17
## YPR149W YPR149W S000006353   NCE102 -4.24   7.95  790 1.16e-19 9.32e-17
## YDR001C YDR001C S000002408     NTH1 -2.89   7.08  650 1.89e-19 1.33e-16
## YHL021C YHL021C S000001013    AIM17 -4.21   6.88  635 3.51e-19 2.19e-16
## YML100W YML100W S000004566     TSL1 -7.12   9.81 1003 5.62e-19 3.15e-16
```

```r
# generate a beautiful table for the pdf/html file.
topTags(res, n=Inf) %>% data.frame() %>%
  arrange(FDR) %>%
  mutate(logFC=round(logFC,2)) %>%
  # mutate(across(where(is.numeric), signif, 3)) %>%
  mutate_if(is.numeric, signif, 3) %>%
  remove_rownames() %>%
  reactable(
    searchable = TRUE,
    showSortable = TRUE,
    columns = list(ORF = colDef(
      cell = function(value) {
        # Render as a link
        url <-
          sprintf("https://www.yeastgenome.org/locus/%s", value)
        htmltools::tags$a(href = url, target = "_blank", as.character(value))
      }
    ))
  )
```

Search

ORF	SGD	GENENAME	logFC	logCPM	F	
YMR105C	S000004711	PGM2	-6.84	9.7	1610	2
YMR196W	S000004809		-5.15	8.36	877	5
YKL035W	S000001518	UGP1	-3.84	10.8	868	5
YDR516C	S000002924	EMI2	-4.01	9.08	795	1
YBR126C	S000000330	TPS1	-3.46	9.81	693	
YLR258W	S000004248	GSY2	-4.86	8.25	680	
YPR149W	S000006353	NCE102	-4.24	7.95	790	1
YDR001C	S000002408	NTH1	-2.89	7.08	650	1
YHL021C	S000001013	AIM17	-4.21	6.88	635	3
YML100W	S000004566	TSL1	-7.12	9.81	1000	5

1–10 of 5615 rows Previous **1** 2 3 4 5 ... 562 Next

```
is.de <- decideTestsDGE(res,
                        p.value=0.05,
```

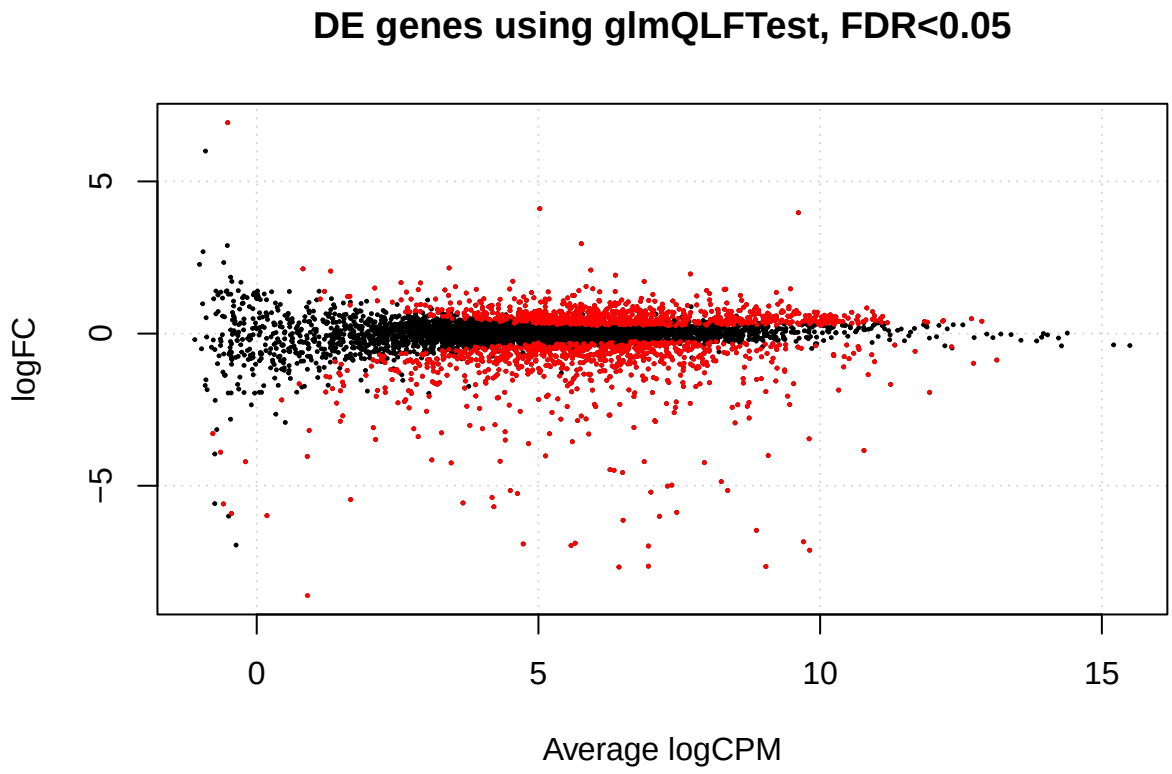
```
                          lfc = 0) # this allows you to set a cutoff, BUT...
# if you want to compare against a FC that isn't 0, should use glmTreat instead.

summary(is.de)
```

```
##        1*WT.unstressed -1*WT.EtOH -1*msn24dd.unstressed 1*msn24dd.EtOH
## Down                                                               761
## NotSig                                                            4031
## Up                                                                 823
```

Let's take a quick look at the differential expression

```
plotSmear(res, de.tags=rownames(res)[is.de!=0])
title(main="DE genes using glmQLFTest, FDR<0.05")
```



Here is how we can save our output file(s).

```
# Choose topTags destination
dir_output_edgeR <-
  path.expand("~/Desktop/Genomic_Data_Analysis/Analysis/edgeR/")
if (!dir.exists(dir_output_edgeR)) {
  dir.create(dir_output_edgeR, recursive = TRUE)
}

# for shairng with others, the topTags output is convenient.
```

```
topTags(res, n = Inf) %>% data.frame() %>%
  arrange(desc(logFC)) %>%
  mutate(logFC = round(logFC, 2)) %>%
  # mutate(across(where(is.numeric), signif, 3)) %>%
  mutate_if(is.numeric, signif, 3) %>%
  write_tsv(., file = paste0(dir_output_edgeR, "yeast_topTags_edgeR.tsv"))

# for subsequent analysis, let's save the res object as an R data object.
saveRDS(object = res, file = paste0(dir_output_edgeR, "yeast_res_edgeR.Rds"))
```

Questions

Question 1: How many genes were upregulated and downregulated in the contrast we looked at in todays activity? Be sure to clarify the cutoffs used for determining significance.

Question 2: Which gene has the lowest pvalue with a postive log2 fold change?

Question 3: Choose one of the contrasts in `my.contrasts` that we didn't test together, and identify the top 3 most differentially expressed genes.

Question 4: In the contrast you chose, give a brief description of the biological interpretation of that contrast.

Question 5: In the example above, we tested for differential expression of any magnitude. Often, we only care about changes of at least a certain magnitude. In this case, we need to use a different command. using the same data, test for genes with differential expression of at least 1 log2 fold change using the `glmTreat` function in edgeR. How do these results compare to DE genes without a logFC cutoff?

A template set of code chunks for doing this is below:

We already loaded in the salmon counts as the object `counts` above. This code chunk just re-downloads that same file.

```
path_salmon_counts <- 'https://github.com/clstacy/GenomicDataAnalysis_Fa23/raw/main/data/ethanol_stress/

counts <- read_tsv(path_salmon_counts,
                   col_names = TRUE) %>%
  column_to_rownames("Name")
```

```
## Rows: 6571 Columns: 17
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (1): Name
## dbl (16): YPS606_MSN24_ETOH_REP1_R1.fastq.gz_quant, YPS606_MSN24_ETOH_REP2_R...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# We are reusing the sample_metadata, group, etc that we assigned above

# create DGEList with salmon counts
y <- DGEList(counts, group=group)
colnames(y) <- sample_metadata$Sample
```

```r
# add gene names
y$genes <- AnnotationDbi::select(org.Sc.sgd.db,keys=rownames(y),
                                          columns="GENENAME")
```

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

```r
# filter low counts
keep <- rowSums(cpm(y) > 60) >= 4
y <- y[keep,]

# calculate norm factors
y <-  calcNormFactors(y)

# estimate dispersion
y <- estimateDisp(y, design, robust=TRUE)

# generate the fit
fit <- glmQLFit(y, design, robust=TRUE)


# Note that, unlike other edgeR functions such as glmLRT and glmQLFTest,
# glmTreat can only accept a single contrast.
# If contrast is a matrix with multiple columns, then only the first column will be used.

# Implement a test against FC at least 1 the test our contrast of interest
tr <- glmTreat(fit,
               contrast = my.contrasts[,"EtOHvsWT.MSN24ddvsWT"],
               lfc=1)

# generate a beautiful table for the pdf/html file.
topTags(tr, n = Inf) %>%
  data.frame() %>%
  arrange(FDR) %>%
  mutate(logFC = round(logFC, 2)) %>%
  # mutate(across(where(is.numeric), signif, 3)) %>%
  mutate_if(is.numeric, signif, 3) %>%
  remove_rownames() %>%
  reactable(
    searchable = TRUE,
    showSortable = TRUE,
    columns = list(ORF = colDef(
      cell = function(value) {
        # Render as a link
        url <-
          sprintf("https://www.yeastgenome.org/locus/%s", value)
        htmltools::tags$a(href = url, target = "_blank", as.character(value))
      }
    ))
  )
```

Search

ORF	SGD	GENENAME	logFC	unshrunk.logFC	logCPM	
YMR105C	S000004711	PGM2	-6.84	-6.84	9.76	8
YML100W	S000004566	TSL1	-7.12	-7.12	9.87	1
YKL035W	S000001518	UGP1	-3.84	-3.84	10.8	3
YMR196W	S000004809		-5.15	-5.15	8.42	
YDR516C	S000002924	EMI2	-4	-4	9.14	2
YPR149W	S000006353	NCE102	-4.24	-4.24	7.94	4
YBR126C	S000000330	TPS1	-3.45	-3.45	9.86	1
YLR258W	S000004248	GSY2	-4.86	-4.86	8.3	3
YFR053C	S000001949	HXK1	-7.65	-7.65	9.1	
YHL021C	S000001013	AIM17	-4.2	-4.2	6.93	1

1–10 of 2372 rows Previous **1** 2 3 4 5 ... 238 Next

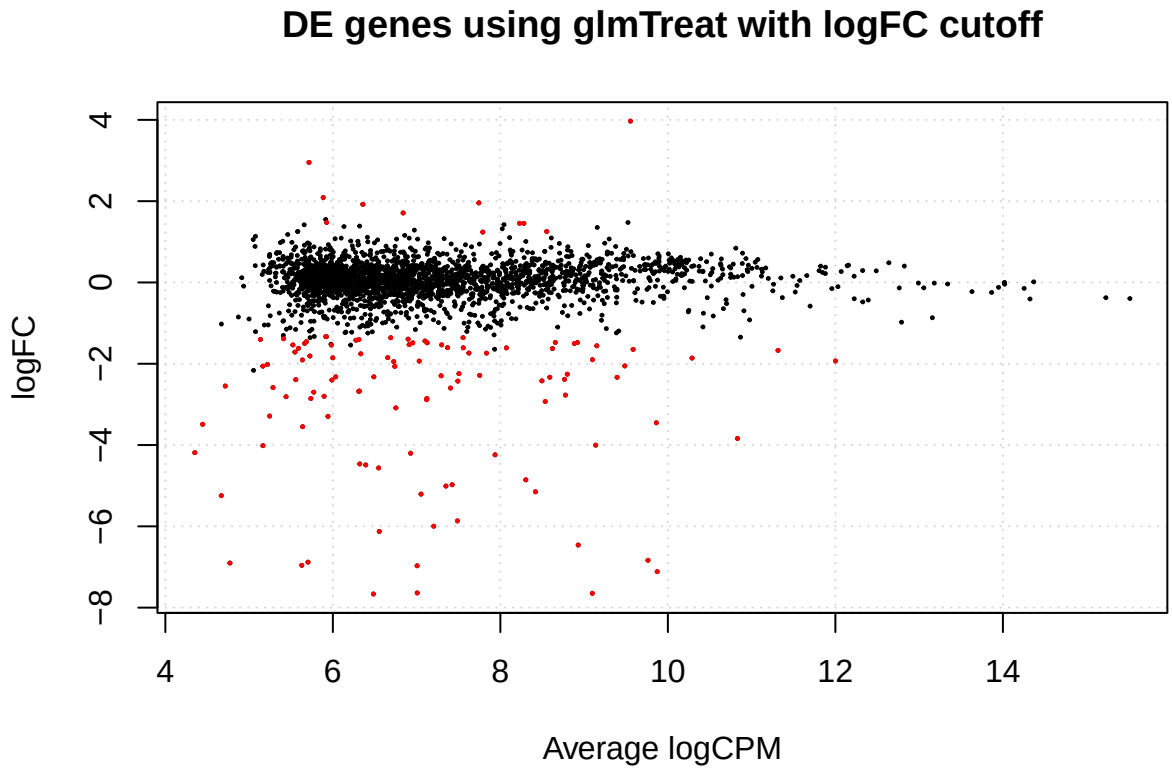
```
# write the table to a tsv file
topTags(tr, n=Inf) %>%
```

```
  data.frame() %>%
  arrange(FDR) %>%
  mutate(logFC=round(logFC,2)) %>%
  # mutate(across(where(is.numeric), signif, 3)) %>%
  mutate_if(is.numeric, signif, 3) %>%
  write_tsv(., file = paste0(dir_output_edgeR, "yeast_lfc1topTags_edgeR.tsv"))

# summarize the DE genes
is.de_tr <- decideTestsDGE(tr, p.value=0.05)
summary(is.de_tr)
```

```
##        1*WT.unstressed -1*WT.EtOH -1*msn24dd.unstressed 1*msn24dd.EtOH
## Down                                                               106
## NotSig                                                            2255
## Up                                                                  11
```

```
# visualize results
plotSmear(tr, de.tags=rownames(tr)[is.de_tr!=0])
title(main="DE genes using glmTreat with logFC cutoff")
```



Be sure to knit this file into a pdf or html file once you're finished.

System information for reproducibility:

```
pander::pander(sessionInfo())
```

R version 4.2.2 (2022-10-31)

Platform: aarch64-apple-darwin20 (64-bit)

locale: en_US.UTF-8||en_US.UTF-8||en_US.UTF-8||C||en_US.UTF-8||en_US.UTF-8

attached base packages: *stats4*, *stats*, *graphics*, *grDevices*, *utils*, *datasets*, *methods* and *base*

other attached packages: *org.Sc.sgd.db(v.3.16.0)*, *AnnotationDbi(v.1.60.2)*, *IRanges(v.2.32.0)*, *S4Vectors(v.0.36.2)*, *Biobase(v.2.58.0)*, *BiocGenerics(v.0.44.0)*, *edgeR(v.3.40.2)*, *limma(v.3.54.2)*, *reactable(v.0.4.4)*, *webshot2(v.0.1.1)*, *statmod(v.1.5.0)*, *BiocManager(v.1.30.21.1)*, *pander(v.0.6.5)*, *knitr(v.1.43)*, *lubridate(v.1.9.2)*, *forcats(v.1.0.0)*, *stringr(v.1.5.0)*, *dplyr(v.1.1.2)*, *purrr(v.1.0.1)*, *readr(v.2.1.4)*, *tidyr(v.1.3.0)*, *tibble(v.3.2.1)*, *ggplot2(v.3.4.2)*, *tidyverse(v.2.0.0)* and *pacman(v.0.5.1)*

loaded via a namespace (and not attached): *bitops(v.1.0-7)*, *bit64(v.4.0.5)*, *webshot(v.0.5.5)*, *httr(v.1.4.6)*, *GenomeInfoDb(v.1.34.9)*, *tools(v.4.2.2)*, *utf8(v.1.2.3)*, *R6(v.2.5.1)*, *DBI(v.1.1.3)*, *colorspace(v.2.1-0)*, *withr(v.2.5.0)*, *tidyselect(v.1.2.0)*, *processx(v.3.8.2)*, *bit(v.4.0.5)*, *curl(v.5.0.1)*, *compiler(v.4.2.2)*, *cli(v.3.6.1)*, *scales(v.1.2.1)*, *digest(v.0.6.33)*, *rmarkdown(v.2.23)*, *XVector(v.0.38.0)*, *pkgconfig(v.2.0.3)*, *htmltools(v.0.5.5)*, *fastmap(v.1.1.1)*, *highr(v.0.10)*, *htmlwidgets(v.1.6.2)*, *rlang(v.1.1.1)*, *rstudioapi(v.0.15.0)*, *RSQLite(v.2.3.1)*, *generics(v.0.1.3)*, *jsonlite(v.1.8.7)*, *crosstalk(v.1.2.0)*, *vroom(v.1.6.3)*, *RCurl(v.1.98-1.12)*, *magrittr(v.2.0.3)*, *GenomeInfoDbData(v.1.2.9)*, *Rcpp(v.1.0.11)*, *munsell(v.0.5.0)*, *fansi(v.1.0.4)*, *lifecycle(v.1.0.3)*, *stringi(v.1.7.12)*, *yaml(v.2.3.7)*, *zlibbioc(v.1.44.0)*, *grid(v.4.2.2)*, *blob(v.1.2.4)*, *parallel(v.4.2.2)*, *promises(v.1.2.0.1)*, *crayon(v.1.5.2)*, *lattice(v.0.21-8)*, *Biostrings(v.2.66.0)*, *splines(v.4.2.2)*, *chromote(v.0.1.2)*, *hms(v.1.1.3)*, *KEGGREST(v.1.38.0)*, *locfit(v.1.5-9.8)*, *ps(v.1.7.5)*, *pillar(v.1.9.0)*, *glue(v.1.6.2)*, *evaluate(v.0.21)*, *png(v.0.1-8)*, *vctrs(v.0.6.3)*, *tzdb(v.0.4.0)*, *gtable(v.0.3.3)*, *reactR(v.0.4.4)*, *cachem(v.1.0.8)*, *xfun(v.0.39)*, *later(v.1.3.1)*, *websocket(v.1.4.1)*, *memoise(v.2.0.1)*, *timechange(v.0.2.0)* and *ellipsis(v.0.3.2)*