RNA-seq workshop

Differential expression analysis in R

Mik Black
Genomics Aotearoa & University of Otago

What are we doing in this session?

- Differential expression analysis in R
- So far we have:
  - QC'd
  - Trimmed
  - Aligned
  - Generated counts per gene
- What is next?
  - use count data to detect differentially expressed genes
  - that involves some statistics....

But first...

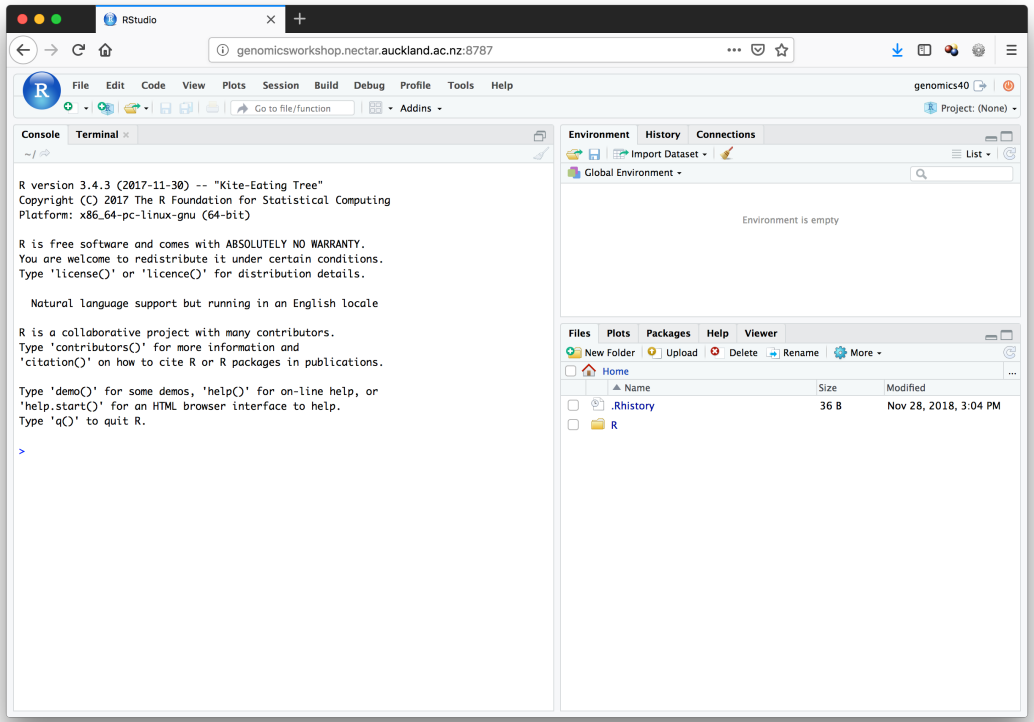
HUGE THANKS TO:

Centre for eResearch
University of Auckland

nectarinfo@auckland.ac.nz

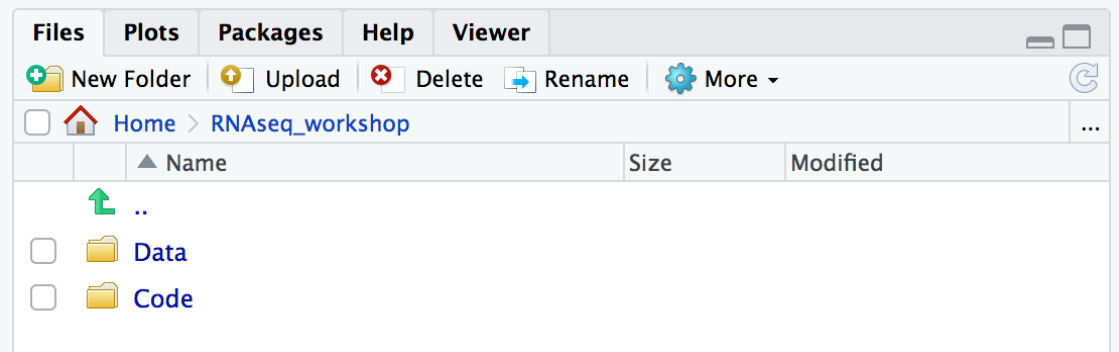
Working in R/RStudio

- Our virtual machine has an instance of RStudio Server running
- Log in at: http://genomicsworkshop.nectar.auckland.ac.nz:8787



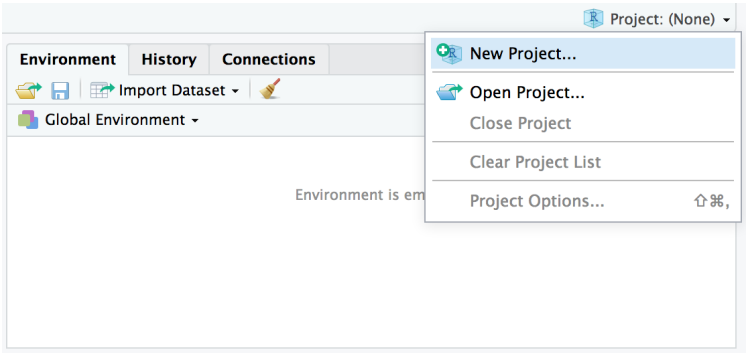
Gettings things set up

- Before we get started, we're going to set up a directory structure to work in, and create an "R project"
- It is good to keep things organised within an analysis project: here we'll set up a project folder called "RNAseq_workshop" using the "New Folder" button in RStudio, and then inside that folder we'll add two additional folders: "Data" and "Code".



Creating a "project"

- RStudio provides functionality to help us keep things organised across analysis projects.
- Use the "Project" menu in the upper right to create a "New Project":



- Choose to use an "Existing Directory" and then select the "RNAseq_workshop"" folder.
- Once you select "Create Project", R will restart, and we will be (almost) ready to do some analysis.

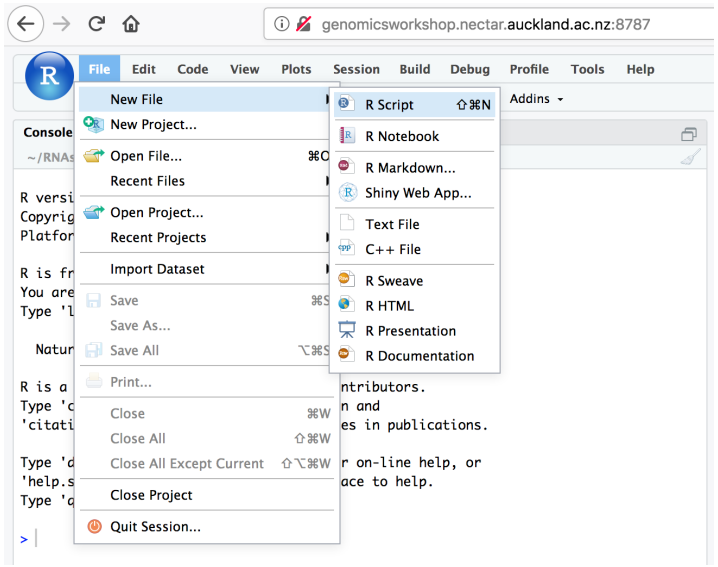
Getting our data

- The only data we need to identify differentially expressed genes is the file of count data that is generated with the `featureCounts` command.
- We need to place this file in the "Data" folder.
- To do this we will use the "terminal" feature of the RStudio interface (access this by clicking the "Terminal" tab next to the "Console" tab) to download the data.
- In the terminal, navigate to the "Data"" directory, and then use the `wget` command to directly download the data from GitHub: the `wget` command and the link to the data can be copied from GitHub:

https://github.com/gregomics/RNAseqWorkshop2018/tree/master/differential_expression

One last thing..

- We are going to save our R commands in a file, in the "Code" folder.
- Create a new file using the "File" menu (choose "New File", "R Script"):



- You can then use "Save As..." (in the "File" menu) to save this file in the "Code" folder.

FeatureCounts

- Let's have a look at the count data:

```
readcounts <- read.table("Data/CountMat_NCBIM37.67.dat", header = TRUE)
```

```
dim(readcounts)
```

```
## [1] 37991    12
```

```
names(readcounts)
```

```
##  [1] "Geneid"  "Chr"     "Start"   "End"     "Strand"  "Length"  "WT1.bam"
##  [8] "WT2.bam" "WT3.bam" "KO1.bam" "KO2.bam" "KO3.bam"
```

- What did we just do?

Viewing the data

```
head(readcounts)[,-c(2:6)]
```

```
##              Geneid WT1.bam WT2.bam WT3.bam KO1.bam KO2.bam KO3.bam
## 1 ENSMUSG00000000702       0       0       0       0       0       0
## 2 ENSMUSG00000078423       0       0       0       0       0       0
## 3 ENSMUSG00000078424       0       0       0       0       0       0
## 4 ENSMUSG00000071964       0       0       0       0       0       0
## 5 ENSMUSG00000093774       0       0       0       0       0       0
## 6 ENSMUSG00000093444       0       0       0       0       0       0
```

- You can also use the `View` command to open the internal viewer
- Best to combine with `head` unless you want to look at the whole file...

```
View( head(readcounts) )
```

Convert to nicer format

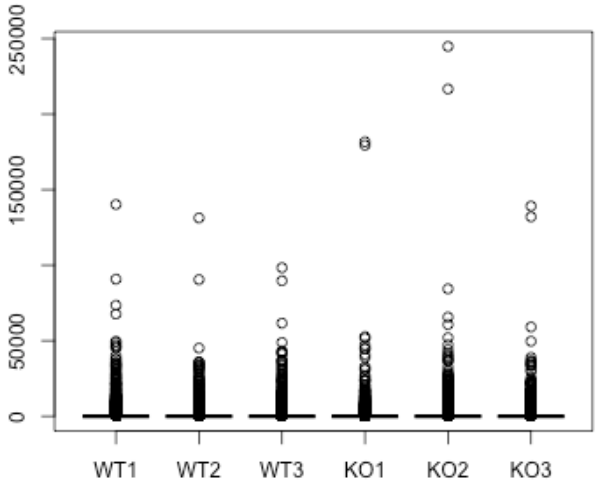
```
## Use the transcript IDs as the row names
row.names(readcounts) <- readcounts$Geneid
## Remove the non-count columns (and convert to a matrix)
counts <- as.matrix(readcounts[ , -c(1:6)])
## Remove the ".bam" suffix from the name of each column
colnames(counts) <- gsub(".bam", "", colnames(counts))
## Show the first 6 rows of the counts object
head(counts)
```

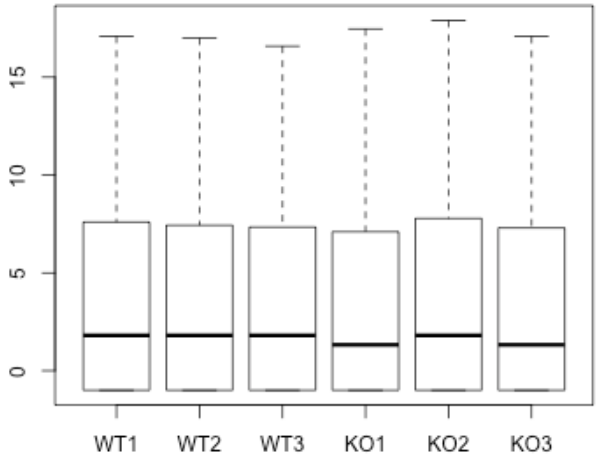
```
##                     WT1 WT2 WT3 KO1 KO2 KO3
## ENSMUSG00000000702   0   0   0   0   0   0
## ENSMUSG00000078423   0   0   0   0   0   0
## ENSMUSG00000078424   0   0   0   0   0   0
## ENSMUSG00000071964   0   0   0   0   0   0
## ENSMUSG00000093774   0   0   0   0   0   0
## ENSMUSG00000093444   0   0   0   0   0   0
```

Visualize the data: boxplots

```
## Take logs of data (add 0.5 to avoid log(0) issues)
logCounts <- log2(counts + 0.5)
```

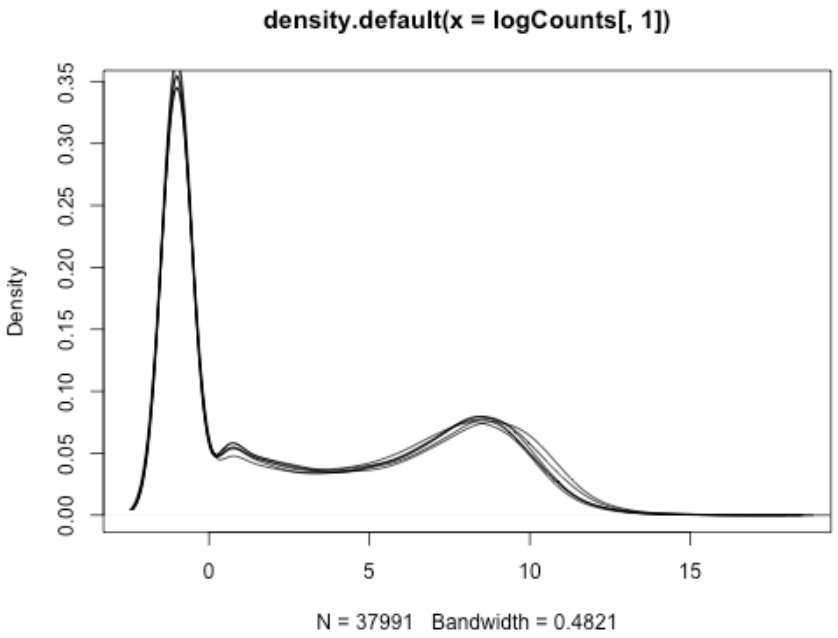
```
## Boxplots of original and logged data
boxplot(counts ~ col(counts), names=colnames(counts))
boxplot(logCounts ~ col(logCounts), names=colnames(counts))
```





Visualize the data: density plots

```
## Plot the density (distribution) for the first sample
plot( density(logCounts[,1]) )
## Use a loop to add the densities for the other samples
for(i in 2:6) lines( density(logCounts[,i]) )
```



Data pre-processing

- Before we can do any statistical analysis, we need to do some "pre-processing" of the data to:
  - make the sure the data are comparible across samples
  - make sure the data are suitable for statistical methods
- We will use the `edgeR` package to help with this (we'll hear more about `edgeR` and `limma` later).

```
library(edgeR)
library(limma)
```

Create a `DGEList` object to work with

```
## Create DGEList object - use counts and gene legnth information
dge <- DGEList(counts=counts, genes=data.frame( length=readcounts$Length) )
names(dge)
```

```
## [1] "counts"  "samples" "genes"
```

```
dge$counts[1:2,]
```

```
##                    WT1 WT2 WT3 KO1 KO2 KO3
## ENSMUSG00000000702   0   0   0   0   0   0
## ENSMUSG00000078423   0   0   0   0   0   0
```

```
dge$genes[1:2,]
```

```
## [1] 1037 1051
```

Create a `DGEList` object to work with

```
dge$samples
```

```
##     group lib.size norm.factors
## WT1     1 12108291            1
## WT2     1  9772751            1
## WT3     1  9855150            1
## KO1     1  8868643            1
## KO2     1 13422084            1
## KO3     1  9638012            1
```

- Note that the `lib.size` info is the total number of reads for each sample.
- This gets used in the preprocessing stage so that the ammout of data for each sample is taken into account.

Define our groups and the "design matrix"

```
groups <- rep(c("WT", "KO"), c(3,3))
groups
```

```
## [1] "WT" "WT" "WT" "KO" "KO" "KO"
```

```
design <- model.matrix(~groups)
design
```

```
##   (Intercept) groupsWT
## 1           1        1
## 2           1        1
## 3           1        1
## 4           1        0
## 5           1        0
## 6           1        0
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$groups
```

Use filtering to remove very low count genes

```
## Figure out what to keep: output is TRUE/FALSE for each gene
keep <- filterByExpr(dge, design)
## Make table of TRUE and FALSE
table(keep)
```

```
## keep
## FALSE  TRUE
## 21651 16340
```

```
## Apply filtering and recalculate library sizes
dge <- dge[keep, keep.lib.sizes=FALSE]
## Calculate normalisation factor (i.e., account for total reads per sample)
dge <- calcNormFactors(dge)
```

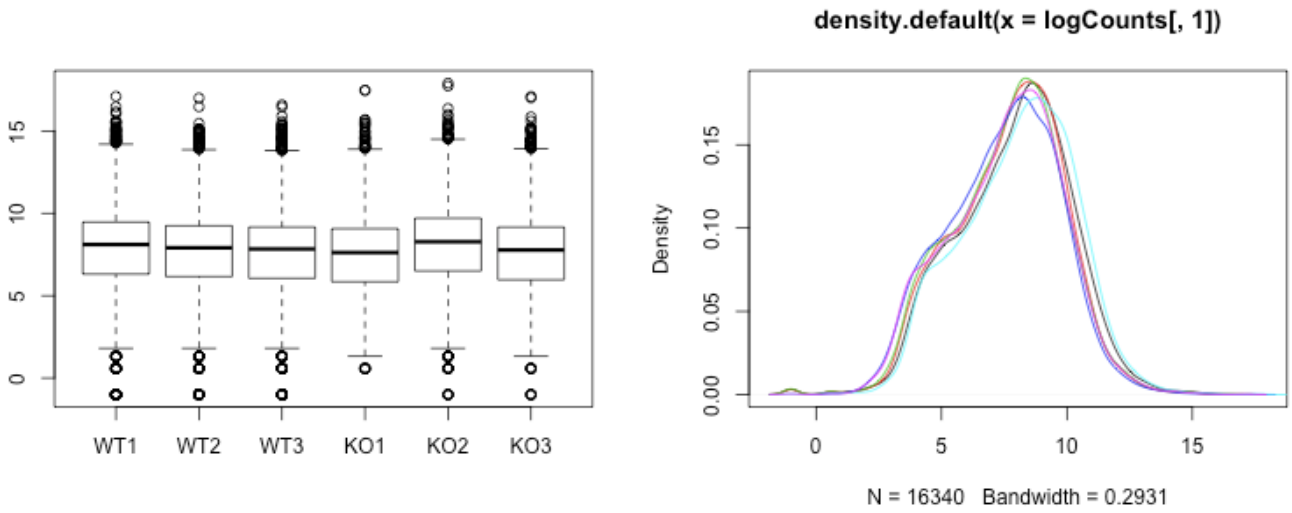
New `DGEList` object

```
dge
```

```
## An object of class "DGEList"
## $counts
##                    WT1 WT2 WT3 KO1 KO2 KO3
## ENSMUSG00000063889  79  69  43  63 104  66
## ENSMUSG00000024231 324 356 299 453 725 369
## ENSMUSG00000024232 389 344 317 174 302 222
## ENSMUSG00000073647  63  95  69 127 182  67
## ENSMUSG00000024235 281 222 177 326 390 148
## 16335 more rows ...
##
## $samples
##     group lib.size norm.factors
## WT1     1 12083144       1.0015
## WT2     1  9752121       1.0494
## WT3     1  9831590       0.9952
## KO1     1  8849450       0.9627
## KO2     1 13398664       0.9938
## KO3     1  9622437       0.9992
```

Visualise the new data

```
## Make log count data
logCounts <- log2(dge$counts + 0.5)
## Boxplots
boxplot(logCounts ~ col(logCounts), names=colnames(logCounts))
## Density plots
plot(density(logCounts[,1]))
for(i in 2:ncol(logCounts)) lines(density(logCounts[,i]), col=i)
```



Counts per million

```
## Generate "counts per million", based on gene lengths
logCPM <- cpm(dge, log=TRUE, prior.count=0.5)
head(logCPM)
```

```
##                        WT1   WT2   WT3    KO1   KO2   KO3
## ENSMUSG00000063889 2.7171 2.763 2.151 2.8956 2.974 2.789
## ENSMUSG00000024231 4.7453 5.122 4.936 5.7338 5.768 5.264
## ENSMUSG00000024232 5.0087 5.073 5.020 4.3555 4.506 4.532
## ENSMUSG00000073647 2.3932 3.222 2.828 3.9024 3.778 2.811
## ENSMUSG00000024235 4.5403 4.442 4.181 5.2597 4.875 3.949
## ENSMUSG00000090484 0.3637 1.001 1.064 0.6536 1.745 1.089
```

Aside: RPKM values

- RPKM stands for "reads per kilobase of exon model per million mapped reads"
- Used to adjust for:
  - total number of reads generated per sample
  - gene length
- Similar measure (FPKM, F='Fragments') used for paired-end data.

$$RPKM = \frac{Reads\ per\ transcript}{\frac{total\ reads}{1{,}000{,}000} \times \frac{transcript\ length}{1000}}$$

$$= \frac{Reads\ per\ transcript}{million\ reads \times transcript\ length\ (kb)}$$

RPKM in R

- Here is how to calculate RPKM values in R, but we are not going to use them.

```
## Use genelength data from dge to calculate rpkm values (not on log scale)
rpkmData <- rpkm(dge)
head(rpkmData)
```

```
##                        WT1     WT2    WT3     KO1    KO2    KO3
## ENSMUSG00000063889 1.0996  1.1356 0.7402  1.2455  1.315 1.1562
## ENSMUSG00000024231 6.0020  7.7978 6.8500 11.9189 12.205 8.6030
## ENSMUSG00000024232 6.3782  6.6693 6.4280  4.0522  4.500 4.5812
## ENSMUSG00000073647 6.9416 12.3770 9.4024 19.8753 18.224 9.2911
## ENSMUSG00000024235 5.1557  4.8162 4.0163  8.4955  6.503 3.4176
## ENSMUSG00000090484 0.3775  0.5951 0.6224  0.4646  1.006 0.6334
```

Analysis with `limma`

- Once the data have been normalized, the `limma` package can be used to detect genes undergoing differential expression.
    - On the log scale this is simply a difference between two means, for each gene/transcript respresented in the pre-processed count data.
- There are many tools available to do this (both in `R`, and elsewhere), but my favourite is the `limma` package.

The limma package

- The `limma` package is available as part of Bioconductor (http://ww.bioconductor.org)
- Developed by Gordon Smyth and colleagues at the Walter and Eliza Hall Institute in Melbourne: http://bioinf.wehi.edu.au/limma/
- Provides tools for the normalization and analysis of of gene expression data: microarrays and RNA-seq data.

Fold changes

- Differential expression (i.e., a change in gene activitation level) is often reported as a *fold change* in activity.
- Often the log_2 scale is used (i.e., log fold change).
- Initially, genes with fold changes greater than 2 ($log_2(2) = 1$) or less than 1/2 ($log_2(\frac{1}{2}) = -1$) were considered to have undergone differential expression.

Detecting differential expression

- How big a change do we need to see for us to think we are observing differential expression? (i.e., what counts as significant differential expression?)
- In order to determine whether a gene has undergone differential expression between two conditions, multiple observations are generally required.
- Assuming that we have multiple expression measurements for a gene under each condition, basic statistical methods can be used to answer this question.

Determining differential expression

- Assume that we want to investigate differences in gene expression between our two cell lines, WT and KO.
- We have three replicates of the WT samples and three replicates of the KO samples.
- For each gene we have 6 data points for each gene (3 in each group, WT and KO).
- For gene k this gives:
  - Group 1 (WT): x_{11}, x_{12}, x_{13}
  - Group 2 (KO): x_{21}, x_{22}, x_{23}

Determining differential expression

- If we assume that all experimental artifacts have been removed by the normalization process, we conclude that any remaining differences in count levels are result of differences in gene expression.
- To test this, we can conduct a formal hypothesis test (for each gene) to determine whether the average count level has changed between the WT and KO samples.
- Since most basic statistical tests are set up to provide answers on the additive scale, and fold changes are on the multiplicative scale, we generally take logs of the data.

Hypothesis testing

- In statistics, we think of our sample means as providing estimates of the underlying (true) population means for each gene, μ_1 and μ_2.
- For each gene, we want to test the following null hypothesis: $H_0 : \mu_1 = \mu_2$ against the alternative hypothesis: $H_A : \mu_1 \neq \mu_2$
- If we reject the null hypothesis for a particular gene, we think that gene is likely to be differentially expressed.

Hypothesis testing

- In order to conduct the hypothesis test, we need a test statistic. A simple approach (but not the best!) is to utilize the test statistic of the standard t-test:

$$T = \frac{\hat{\mu}_1 - \hat{\mu}_2}{SE(\hat{\mu}_1 - \hat{\mu}_2)}$$

where $\hat{\mu}_1$ and $\hat{\mu}_2$ are the sample means of the data, and $SE(\hat{\mu}_1 - \hat{\mu}_2)$ is some appropriate measure of variability (in this case the *standard error*).

- Various choices are possible for the denominator depending on the structure of the data.

Two sample t-test in R

```
## The function "t.test"" performs a two sample t-test in R.
## Perform t-test on first gene in data set (first row of matrix:
## first three values are WT, second three are KO):
t.test(logCPM[1,] ~ groups)
```

```
##
##  Welch Two Sample t-test
##
## data:  logCPM[1, ] by groups
## t = 1.7, df = 2.3, p-value = 0.2
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.4355  1.1202
## sample estimates:
## mean in group KO mean in group WT
##            2.886            2.544
```

Two sample t-test in R

```
## Very first gene does not appear to be
## differentially expressed between the two groups:
t.test(logCPM[1,]~groups)$p.value
```

```
## [1] 0.219
```

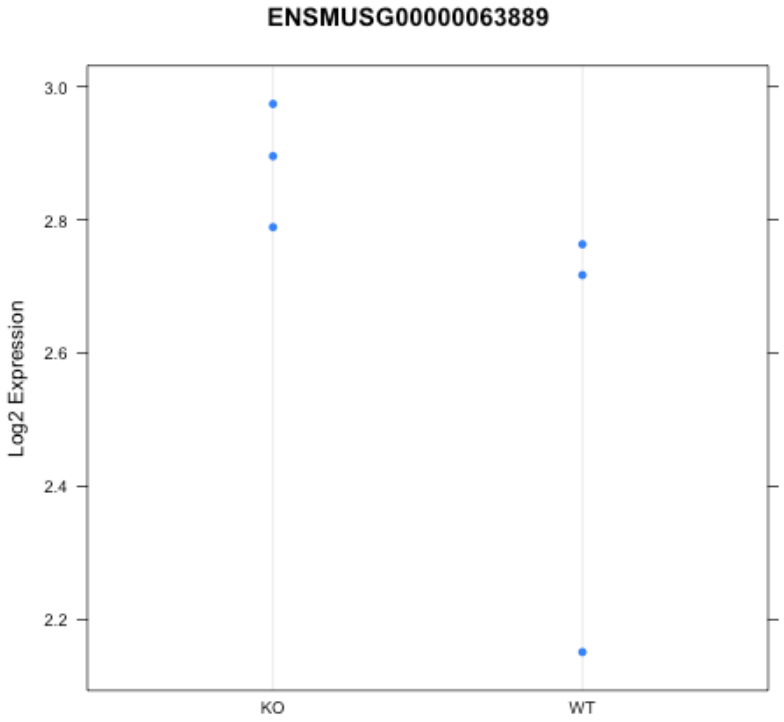
```
## How big is the change?
2^(2.886-2.544)
```

```
## [1] 1.268
```

- Expression is 27% higher in the KO samples relative to the WT samples.
- Our statistical test tells us that this is not a significant result.

Visualize the data

```
## Examine expression graphically using a dotplot:
lattice::dotplot(logCPM[1, ] ~ groups, main = rownames(logCPM)[1], ylab = "Log2 Expression")
```



P-values

- Once we have calculated a gene-specific test statistic, we can either use a *t* distribution, or resampling methods (e.g., randomization test) to calculate a *p*-value for each gene, p_k.
- The *p*-value represents the probability of observing this (or a more extreme) result, if no differential expression occurred. (i.e., what is the chance we are just observing noise?)
- We reject H_{0k} (i.e., say gene k is differentially expressed) if p_k is small.
- Question: what does small mean?

P-values

- We have to decide how small a *p*-value needs to be for us to think that the difference we are observing cannot be explained solely by noise.
- When we test a single hypothesis, it is common to fix a Type I error rate of α= 0.05 or α = 0.01.
- Type I error: reject null hypothesis when it is true (i.e., say a gene is differentially expressed when it really isn't).
- Type II error: fail to reject the null hypothesis when it is false (i.e., say a gene is not differentially expressed when it really is).

Type I errors

- Using a Type I error rate of α= 0.05 means that we are willing to make a Type I error in 5% of our hypothesis tests (i.e., 5% of the time that the null hypothesis is true, we will say that it's false).
- So for every 20 hypothesis tests we perform, on average we expect 1 Type I error.
- What if we are performing 20,000 hypothesis tests?

1000 TYPE I ERRORS!

Adjusting the α level

- Obviously using an α level of 0.05 (or even 0.01) is not suitable when testing large numbers of hypotheses.
- To get around this problem we use Multiple Comparison Procedures (MCPs).
- MCPs provide error rate control, allowing us to keep a lid on how many Type I errors we make.

Family-wise error rate control

- Control of the family-wise error rate (FWER) is very common in multiple testing problems.
- MCPs which control the FWER guarantee that the FWER $< \alpha$, where a "family-wise error" is defined to be the occurrence of a single Type I error in the entire family (set) of hypotheses being tested.
- In a transcriptomic experiment we test each gene for differential expression, so there are as many hypothesis tests as there are genes.
- The Bonferroni and Holm procedures both provide control of the FWER.

What's so great about FWER control?

- Advantage: FWER controlling procedures provide a high level of certainty in your result. The null hypotheses rejected by these procedures are very unlikely to be true (i.e., all of the rejected null hypotheses are likely to be correct rejections).
- Disadvantages: This level of control is very conservative - it is likely that some genes undergo differential expression, but their null hypotheses are not rejected. As the number of hypotheses being tested becomes very large, the significance threshold becomes extremely small.

What is the alternative?

- Continue to control the FWER, but use a larger value?
- Switch to a different error rate?
- What other error rates exist? (not many...)

False Discovery Rate control

- The False Discovery Rate was introduced by Benjamini and Hochberg (1995 - JRSS(B)).
- Provides a less conservative approach to error rate control than FWER controlling procedures.
- Greater power comes at the cost of an increased likelihood of Type I errors.
- Has become very popular in genomic data analysis, plus astronomy, brain imaging, and genetics (all test large numbers of hypotheses).

FDR control versus FWER control

- FWER control is concerned with making sure that the probability of a single testing error is small.
- FDR control in concerned with keeping the proportion of Type I errors out of the total number of rejected hypotheses small.
  - This value can be anything from 0 to 1.
- FDR controlling procedures provide more error rate protection than not adjusting at all, but are a lot more likely to make Type I errors than FWER controlling procedures.
- The flip side is that FDR controlling methods are more likely to reject false null hypotheses (i.e., they achieve greater power).

Comparing approaches

- Instead of adjusting the signifiance threshold, we can adjust the p-values themselves.
- The table below contains unadjusted p-values ("P-value"), and p-values adjusted using the Bonferroni, Holm, and FDR methods.
- For α=0.05, the four approaches find 7, 2, 4, or 6 tests significant.

TEST NUMBER	P-VALUE	BONFERRONI	HOLM	FDR
1	0.002	0.016	0.016	0.0160
2	0.004	0.032	0.028	0.0160
3	0.007	0.056	0.042	0.0187
4	0.010	0.080	0.050	0.0200
5	0.020	0.160	0.080	0.0320
6	0.030	0.240	0.090	0.0400
7	0.050	0.400	0.100	0.0571
8	0.080	0.640	0.100	0.0800

Multiple t-tests for transcriptome data

- P-values for multiple genes (first 10).

```
pvalues = c()
for(i in 1:10) pvalues[i] = t.test( logCPM[i,] ~ groups )$p.value
sort( round(pvalues,4) )
```

```
##  [1] 0.0047 0.0353 0.1862 0.1880 0.2190 0.4202 0.4482 0.4999 0.5177 0.5758
```

Multiple t-tests for transcriptomic data

- Use `p.adjust` to apply correction (default is Holm):

```
round(p.adjust(sort(pvalues)),4)
```

```
##  [1] 0.0469 0.3177 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
```

```
round(p.adjust(sort(pvalues),"BH"),4)  # BH is the FDR method
```

```
##  [1] 0.0469 0.1765 0.4380 0.4380 0.4380 0.5752 0.5752 0.5752 0.5752 0.5758
```

- So here the test with the smallest p-value remains (just) significant, but the remainder do not.

Detecting differential expression

- Steps in statistical analysis

    1. Background correction/Normalization.
    2. Assess chance of differential expression for each gene.
    3. Determine significance by controlling Type I error rate.

- We've covered 1 and 3 in reasonable detail, what about step 2?

Modification to t-test procedure

- One problem with the t-statistic approach to determining significance is that some genes with small, but consistent fold changes can end up with very large t-statistics.
- This is especially common is experiments involving only a few samples.
- Generally feel that genes with small fold changes shouldn't be considered as having undergone significant differential expression.
- Need a way to prevent these genes showing up as significant.

Significance Analysis for Microarrays (SAM)

- Tusher et al. (2001) proposed a modification to the denominator of the t-statistic to reduce the influence of tiny standard deviations.

$$T = \frac{\hat{\mu}_1 - \hat{\mu}_2}{SE(\hat{\mu}_1 - \hat{\mu}_2) + s_0}$$

- Although this modification looks somewhat arbitrary, it can be derived by taking a Bayesian approach to analysis (and various other ways).
- The Bayesian derivation relies on the assumption that the standard errors for each gene have an underlying common distribution. The s_0 parameter then contains information from this underlying distribution.

Significance Analysis for Microarrays (SAM)

- Although simple, this approach is highly effective, and has become a popular method for detecting genes undergoing significant differential expression.
- Various methods can be employed for estimating the s_0 parameter.
- Tusher et al. (2001) chose s_0 to minimize the coefficient of variation.
- Other authors have suggested using quantiles of the underlying empirical (observed) distribution of standard errors (much easier).

Significance Analysis for Microarrays (SAM)

- Has the effect of restricting significant genes to those exhibiting large fold changes.
- Although the distribution of T is unknown, resampling methods (e.g., bootstrapping) can be used to approximate the null distribution, allowing calculation of *p*-values.
- Multiple comparison procedures can then be used to provide control of the Type I error rate (FWER or FDR).

Detecting differential expression with `limma`

- The `limma` package takes a linear models approach to detecting genes which have undergone differential expression.
- After the data have been normalized, a linear model is fit to the expression values to determine which genes underwent significant changes.
- Although a standard t statistic can be used to assess differential expression, limma goes a little bit further...

WARNING:

The next few slides may hurt your brain...

Empirical Bayes analysis

- Limma uses Empirical Bayes methods to produce a modified test statistic.
- The idea is similar to that employed by the SAM procedure, but is more sophisticated, and has more solid mathematical foundations.
- The goal is to modify the denominator of a standard t test statistic, by making large standard errors smaller, and small standard errors larger.
- This is known as *shrinkage estimation*.

Shrinkage estimation

- The underlying assumption is that the gene-specific variances follow a standard distribution (e.g., a gamma distribution) with some fixed parameters.
- This provides us which information about the underlying spread of the gene-specific variances.
- When we see extreme values from this distribution, we would like to *moderate* them, so that they don't have a major effect on our results (i.e., want to make large standard errors smaller, and small standard errors larger).

Shrinkage estimation

- To accomplish this, a weighted variance is calculated, based on the observed gene-specific variance, and the characteristics of the underlying distribution.
- This has the effect of pulling the extreme value towards the centre of the observed (empirical) distribution of gene-specific variances.

Why is it empirical Bayes?

- The procedure is considered Bayesian because by assuming an underlying distribution, we are effectively adding *a priori* knowledge to our problem by imposing a prior distribution on the gene-specific variances.
- This particular approach is *empirical* Bayes because it uses the data from the empirical (observed) distribution of gene-specific variances to estimate the parameters of the prior distribution.

Back to limma

- Once limma has fit a linear model to the normalized data (using `lmFit`), a second function (`eBayes`) is used to calculate *moderated t-statistics* based on shrunken estimates of the per-gene variances.
- The moderated t-statistics can be quite different than the standard t-statistics, especially for small sample sizes.
- In general, the moderated t-statistics make it more likely that significant genes will have a large fold change, and a small variance, rather than a small fold-change and a tiny variance.

Determining differential expression

- Because of the mathematics underpinning the empirical Bayes approach, the moderated t-statistics still follow a standard t-distribution (unlike the SAM approach), with degrees of freedom based on both the number of observations for each gene, and the parameters of the underlying prior distribution.
- This allows the calculation of parametric p-values, to which standard multiple comparisons procedures can be applied.

Background: linear models

- Simple linear regression: $y = mx + b$
- Linear model equivalent: $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$
- In linear regression, x and y are continuous variables. Here we have y (gene expression) as continuous, but x (group) is discrete, so our linear model is actually equivalent to ANOVA (analysis of variance).
- For a single gene:
  - y_i are our gene expression values
  - x_i is the group (WT or KO) for the i^{th} sample
  - β_0 and β_1 are the intercept and slope coefficients
  - ϵ_i is the residual (or error) associated with obsevation y_i (the difference between our predicted, $\hat{y}_i$ and observed, y_i, values that cannot be explained by the model).

Background: linear algebra

- In practice, we represent our linear model in matrix form:

$$Y = X\beta + \epsilon$$

and use basic linear algebra to solve the equation and determine the value of the coefficients.

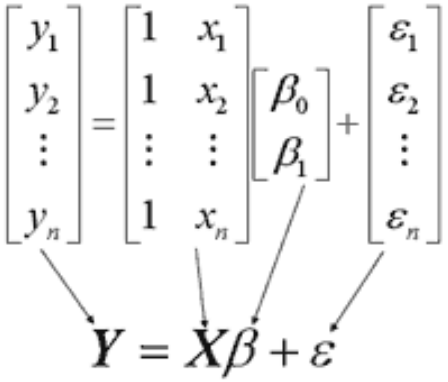


Image from: https://onlinecourses.science.psu.edu/stat501/node/382

Background: linear algebra

- The solution that minimises the "sums of squared error":

$$\sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

  is given by:

$$\hat{\beta} = (X'X)^{-1}X'Y$$

- Why do we care?
  - Because `limma` requires the *design matrix*, X, to fit this model *per gene* and estimate its probability of differential expression.

Okay, brains back on...

The design matrix

- Remember our design matrix:

```
design
```

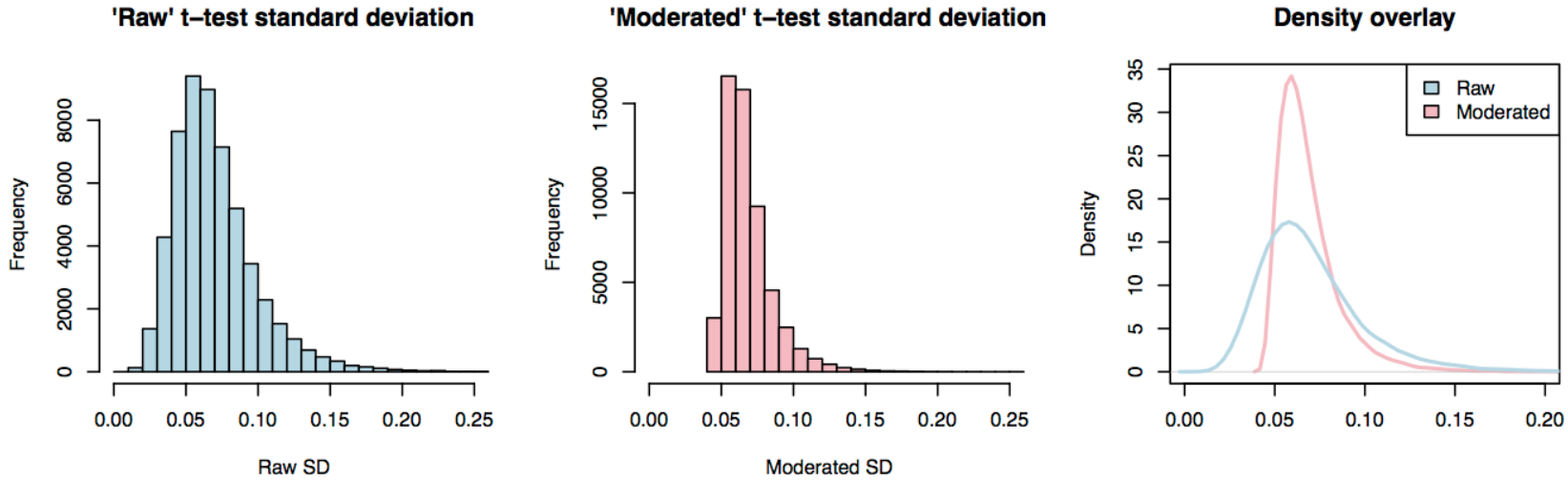
```
##   (Intercept) groupsWT
## 1           1        1
## 2           1        1
## 3           1        1
## 4           1        0
## 5           1        0
## 6           1        0
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$groups
## [1] "contr.treatment"
```

The design matrix

- For our simple two-group differential expression analysis, the design matrix has two columns:
  - the first is all ones, and relates to the intercept coefficient: it is the average level of log-expression for the gene (remember the linear model is fit to each gene, so we have an intercept and a "slope" term *per gene*),
  - the second has zeroes for one group, and ones for the other, and relates to the coefficient for group ("slope"): it is the avearge difference in log-expression between the groups for that gene). *This is what we are interested in*.
- The residuals (the ϵ_i's) for each gene are used to determine whether the observed expression difference is statisically significant.

And speaking of residuals...

- This is where `limma` "moderates" the denominator of the t-statistic.



- The moderated SD distribution (pink) is *shrunk* towards the center: large SDs get (slightly) smaller, and *small SDs get larger*, so large t-statistics become smaller.

Determining differential expression

- Limma also reports the *log odds* of differential expression.
- This quantity has a more Bayesian "feel" to it, providing a measure of how likely it is for a gene to have undergone differential expression, relative to the null hypothesis of no differential expression.
- Genes with high (positive) log odds are considered likely to have undergone differential expression.

Limma: detecting differential expression

```
## Load the limma package
library(limma)
## Fit linear model
fit = lmFit(logCPM, design)
fit = eBayes(fit)
tt = topTable(fit, coef=2, adjust="BH", n=nrow(logCPM))
options(digits=4)
tt[1:5,]
```

```
##                     logFC AveExpr      t   P.Value adj.P.Val     B
## ENSMUSG00000079112 -7.019 -0.8959 -53.32 2.245e-10 2.130e-06 12.16
## ENSMUSG00000086445 -7.494 -0.6582 -52.19 2.607e-10 2.130e-06 12.10
## ENSMUSG00000087318 -6.514 -1.1483 -39.51 1.807e-09 7.543e-06 11.25
## ENSMUSG00000020673 -7.560 -0.6252 -39.39 1.846e-09 7.543e-06 11.24
## ENSMUSG00000085973 -6.205 -1.3030 -30.77 1.026e-08 3.353e-05 10.23
```

Check output against t-test (top gene)

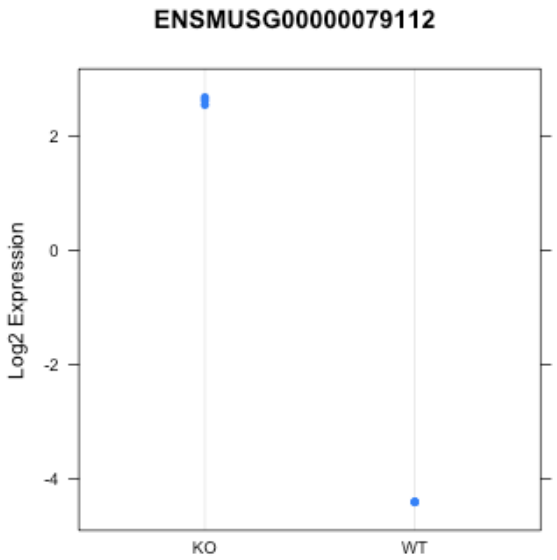
```
topGene = match(rownames(tt)[1], rownames(logCPM))
t.test(logCPM[topGene,] ~ groups)
```

```
##
##  Welch Two Sample t-test
##
## data:  logCPM[topGene, ] by groups
## t = 180, df = 2, p-value = 3e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  6.853 7.184
## sample estimates:
## mean in group KO mean in group WT
##            2.613           -4.405
```

```
# "Raw" data agrees with logFC (WT - KO): -4.405 - 2.613 = -7.018
```

Dotplot of expression (top gene)

```
lattice::dotplot(logCPM[topGene,] ~ groups, main=rownames(logCPM)[topGene],
           ylab='Log2 Expression')
```



Clearly there is a major difference in expression between the WT and KO samples for this gene.

Limma: detecting differential expression

```
sum( tt$adj.P.Val < 0.05 )
```

```
## [1] 598
```

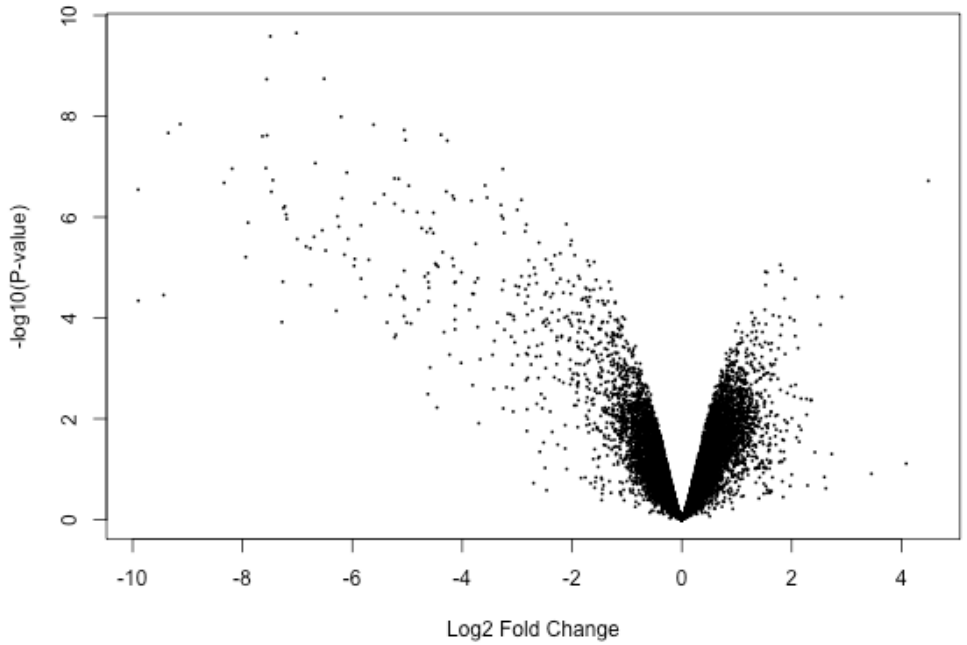
598 transcripts found to be differentially expressed using the False Discovery Rate (FDR) criterion.

Get the rows for the significant genes (we'll use these later):

```
limmaPadj <- tt[tt$adj.P.Val <= 0.05, ]
```

Volcano Plot

```
## Plot log fold-change _versus_ -log(P-value)
## (i.e., higher number = lower p-value):
volcanoplot(fit, coef=2)
```



- Volcanoplots are a useful tool for visualising the results of a differential expression analysis.
- Easy to identify significant changes that are also large in magnitude.

Annotation: additional information

- From wikipedia.org: "Annotation is extra information associated with a particular point in a document or other piece of information."
- Here our "document" is the genome.
- The goal of annotating the genome is to link *all* information relating to sequences, genes, protein, function...

Entrez Gene

- When a genome is an annotated, genes are assigned an "Entrez gene" (EG) identifier.
- The gene identifier is also linked to a more descriptive gene name. This usually conveys some information about what that gene does (or at least what it was understood to be involved in at the time it was named).
- Depending on what is known about these genes (e.g., via information from previous experiments, or sequence similarity with genes in other organisms), this information may provide important clues about the underlying biological process being studied.

Which genes are significant?

```
## Load Entrez-centric annotation for Mouse (Mus musculus: Mm)
library(org.Mm.eg.db)
## Get the IDs for the significant transcripts
sigGenes <- rownames(limmaPadj)
## Get the gene symbols and names associated with these transcripts (first 6, via head)
options(width=100)
select(org.Mm.eg.db, keys = head(sigGenes), column = c("SYMBOL", "GENENAME"),
       keytype="ENSEMBL")
```

```
##              ENSEMBL   SYMBOL                                              GENENAME
## 1 ENSMUSG00000079112 Fam90a1a family with sequence similarity 90, member A1A
## 2 ENSMUSG00000086445     <NA>                                              <NA>
## 3 ENSMUSG00000087318     <NA>                                              <NA>
## 4 ENSMUSG00000020673      Tpo                                 thyroid peroxidase
## 5 ENSMUSG00000085973     <NA>                                              <NA>
## 6 ENSMUSG00000043110    Lrrn4                      leucine rich repeat neuronal 4
```

List of significant genes

```
allSigGenes <- select(org.Mm.eg.db, keys = sigGenes, column = "SYMBOL",
                      keytype="ENSEMBL")[,2]
```

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

```
na.omit(allSigGenes)
```

```
##   [1] "Fam90a1a"      "Tpo"           "Lrrn4"         "Iqcf5"         "Hnf4aos"
##   [6] "Chrm3"         "Sp110"         "Fbxo16"        "Dhtkd1"        "Olfr815"
##  [11] "Slc14a2"       "Prr18"         "AI427809"      "Krt2"          "Abcb5"
##  [16] "Tie1"          "Echdc3"        "Mecom"         "Cryzl2"        "Ccdc60"
##  [21] "Loxhd1"        "Aqp9"          "4930432K21Rik" "Oas1a"         "Artn"
##  [26] "H2-M10.4"      "Ccser1"        "A530032D15Rik" "LOC100041708"  "Apol7b"
##  [31] "Stox1"         "Slfn2"         "BB557941"      "Rasd2"         "Dlgap2"
##  [36] "Oas1g"         "0610040J01Rik" "Defb13"        "Supt3"         "Colgalt2"
##  [41] "B3gntl1"       "Lmx1a"         "Dcc"           "Acsf2"         "Iars"
##  [46] "Sp140"         "Chit1"         "Kcnk15"        "Phf11a"        "Prelid2"
##  [51] "Mcm9"          "Apol9a"        "Arg2"          "Ahrr"          "AU019990"
##  [56] "Arhgap8"       "Spdye4b"       "H2-T10"        "Arc"           "Cd207"
```

Gene function

- Usually it's not particularly interesting to find out that a gene is significantly differentially expressed if no other information is known about that gene.
- One (very good) reason for this is that in transcriptomic experiments there are often a lot of false positives, so we tend to get a little bit skeptical...
- Remember: we can only have as much faith in the analysis as we do in the underlying assumptions. Were those genes REALLY independent? How about the residuals - normally distributed?

Gene function

- If enough is known about a differentially expressed gene for it to "make sense" or be "interesting" in the context of the experiment, then we tend to get a bit more excited.
- Although a gene name is often somewhat informative, vast amounts of information about that gene may reside in journal publications and internet databases - how do we get this information?

PubMed identifiers

- PubMed is a service provided by the National Library of Medicine.
- Contains over 28 million citations from MEDLINE and other life sciences publications.
- Every journal publication is given a unique PubMed identifier.
- Those that relate to a particular gene or sequence are linked back to the appropriate identifiers.
- Based on this the NCBI search engine hosts a local copy of the NCBI databases) can be used to retrieve information about differentially expressed genes.

Problem - too much information

- For situations where large numbers of genes are differentially expressed, there is simply too much information available.
- Anyway, are we really interested in individual genes?
- Wouldn't it be better to find groups of differentially expressed genes which share a common function?

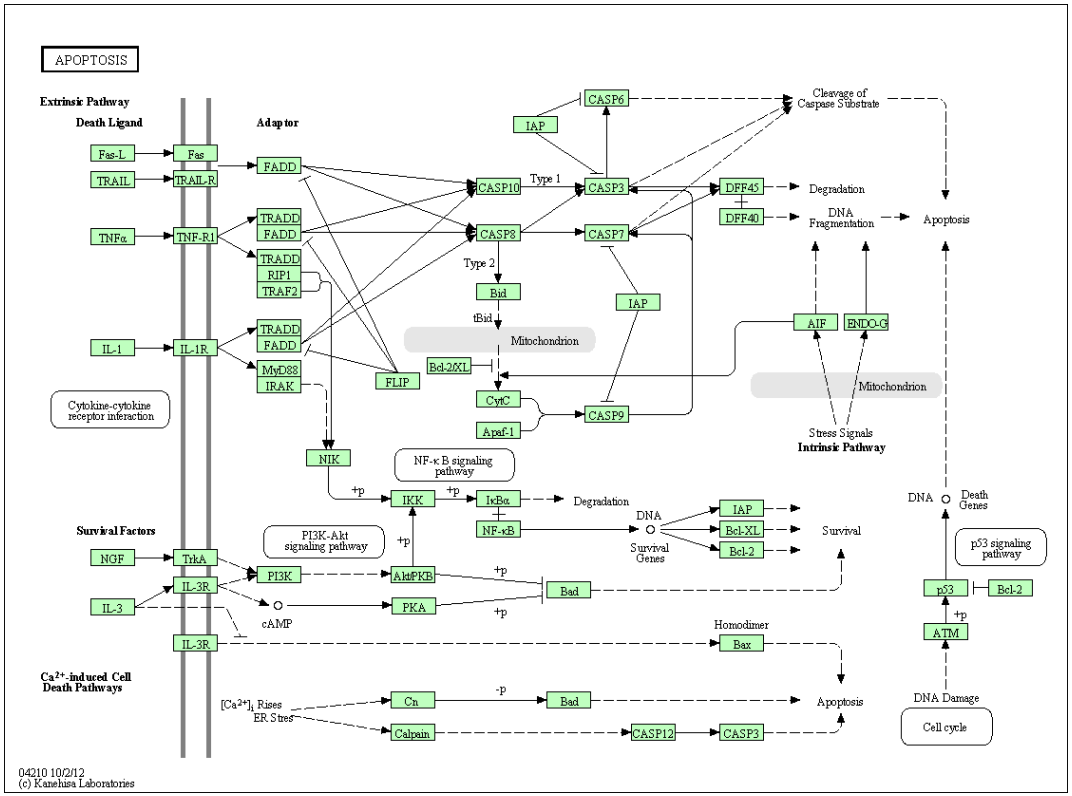
Biological pathways

- In reality genes are members of *pathways*, which perform major biological functions.
- As more biological experimentation is done, researchers are able to build a better picture of how genes interact, and how pathways function.
- Information about pathway membership and gene function are stored in publicly available databases.
- This information can be used to define *gene sets* (groups of genes which are functionally related), to which statistical analysis can be applied.

Biological pathways: KEGG

- Kyoto Encyclopedia of Gene and Genomes:
  http://www.genome.jp/kegg/kegg4.html
- Provides nice (user-created) pathway diagrams.
- XML output includes information about genes involved in pathways, and inter-gene (and gene product) relationships.
  - Can produce graphic representation of pathway based on XML alone.
- Access to database now requires a subscription (can still view pathway maps for free though).

KEGG pathway diagram (apoptosis)



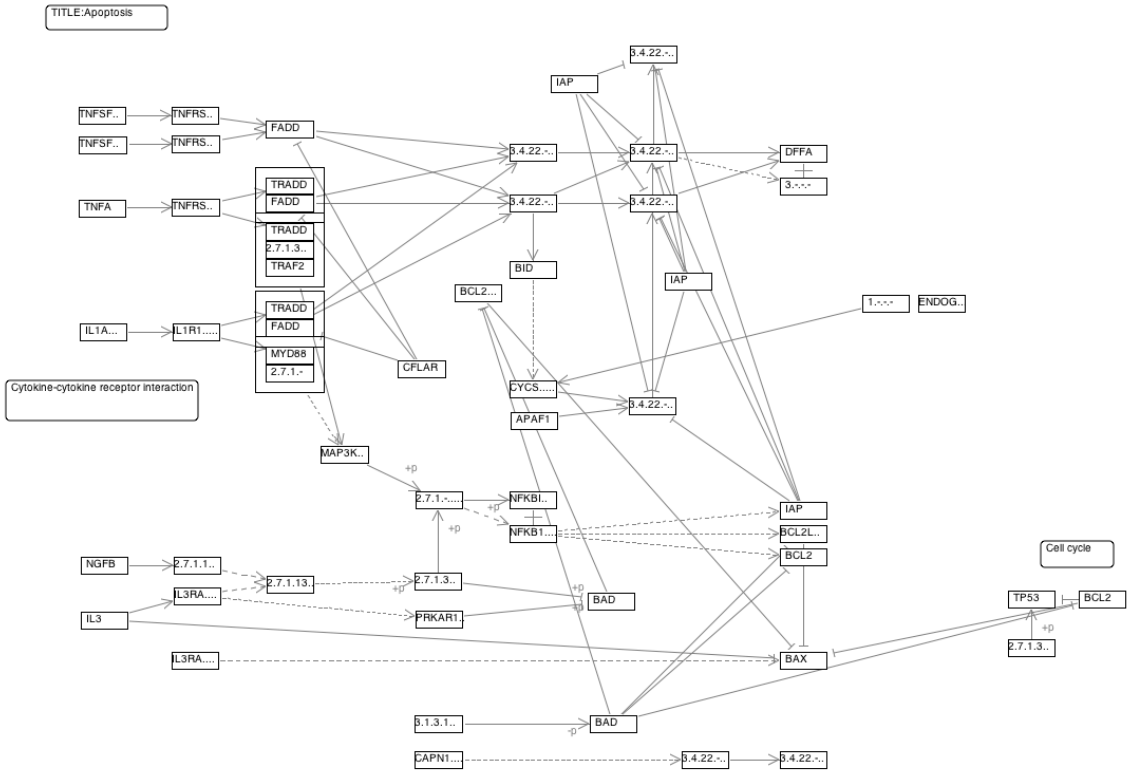
http://www.genome.jp/kegg-bin/show_pathway?org_name=hsa&mapno=04210&mapscale=&show_description=show

XML output file for apoptosis pathway

```
:<?xml version="1.0"?>
<!DOCTYPE pathway SYSTEM "http://www.kegg.jp/kegg/xml/KGML_v0.7.1_.dtd">
<!-- Creation date: Oct 2, 2012 11:48:00 +0900 (GMT+09:00) -->
<pathway name="path:hsa04210" org="hsa" number="04210"
         title="Apoptosis"
         image="http://www.kegg.jp/kegg/pathway/hsa/hsa04210.png"
         link="http://www.kegg.jp/kegg-bin/show_pathway?hsa04210">
    <entry id="1" name="path:hsa04115" type="map"
        link="http://www.kegg.jp/dbget-bin/www_bget?hsa04115">
        <graphics name="p53 signaling pathway" fgcolor="#000000" bgcolor="#FFFFFF"
            type="roundrectangle" x="1049" y="572" width="95" height="39"/>
    </entry>
    <entry id="2" name="path:hsa04060" type="map"
        link="http://www.kegg.jp/dbget-bin/www_bget?hsa04060">
        <graphics name="Cytokine-cytokine receptor interaction" fgcolor="#000000" bgcolor="#FFFFFF"
            type="roundrectangle" x="111" y="427" width="124" height="39"/>
    </entry>
    <entry id="3" name="hsa:5530 hsa:5532 hsa:5533 hsa:5534 hsa:5535" type="gene"
        link="http://www.kegg.jp/dbget-bin/www_bget?hsa:5530+hsa:5532+hsa:5533+hsa:5534+hsa:5535">
        <graphics name="PPP3CA, CALN, CALNA, CALNA1, CCN1, CNA1, PPP2B..." fgcolor="#000000" bgcolor="#BFFFBF"
            type="rectangle" x="430" y="733" width="46" height="17"/>
    </entry>
    <entry id="4" name="hsa:581" type="gene"
        link="http://www.kegg.jp/dbget-bin/www_bget?hsa:581">
        <graphics name="BAX, BCL2L4" fgcolor="#000000" bgcolor="#BFFFBF"
            type="rectangle" x="776" y="673" width="46" height="17"/>
    </entry>
    <entry id="5" name="hsa:598" type="gene"
        link="http://www.kegg.jp/dbget-bin/www_bget?hsa:598">
        <graphics name="BCL2L1, BCL-XL/S, BCL2L, BCLX, BCLXL, BCLXS, Bcl-X, PPP1R52, bcl-xL, bcl-xS" fgcolor="#000000" bgcolor="#BFFFBF"
            type="rectangle" x="776" y="553" width="46" height="17"/>
    </entry>
    <entry id="6" name="hsa:1676" type="gene"
        link="http://www.kegg.jp/dbget-bin/www_bget?hsa:1676">
        <graphics name="DFFA, DFF-45, DFF1, ICAD" fgcolor="#000000" bgcolor="#BFFFBF"
            type="rectangle" x="776" y="192" width="46" height="17"/>
    </entry>
    <entry id="7" name="hsa:596" type="gene"
        link="http://www.kegg.jp/dbget-bin/www_bget?hsa:596">
        <graphics name="BCL2, Bcl-2, PPP1R50" fgcolor="#000000" bgcolor="#BFFFBF"
            type="rectangle" x="1059" y="615" width="46" height="17"/>
    </entry>
    <entry id="8" name="hsa:472" type="gene"
        link="http://www.kegg.jp/dbget-bin/www_bget?hsa:472">
```

http://www.kegg.jp/kegg-bin/download?entry=hsa04210&format=kgml

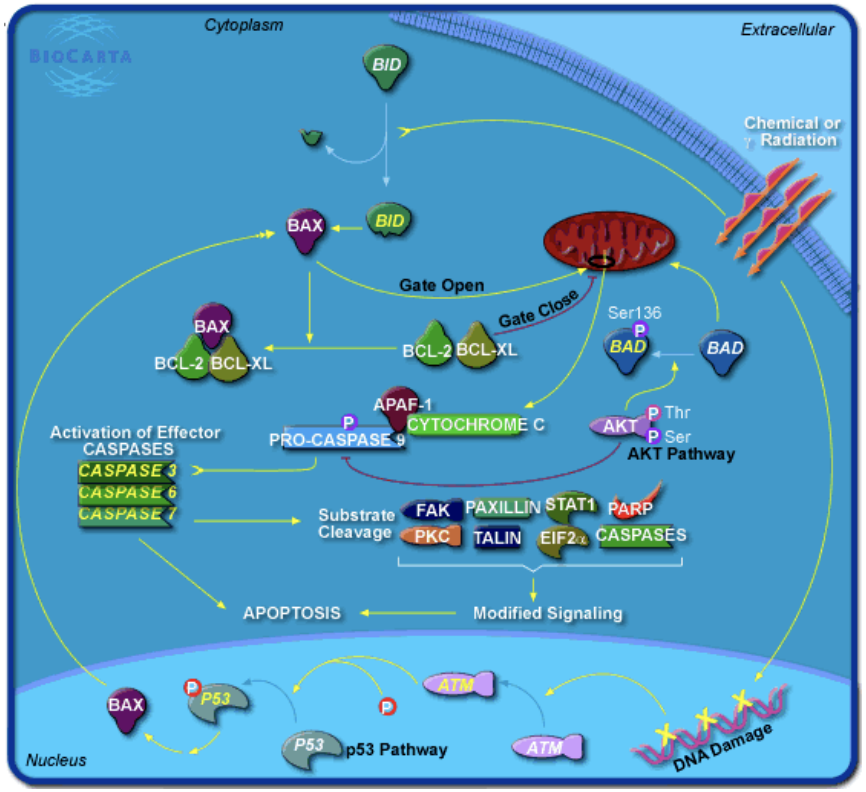
XML-based KEGG diagram (apoptosis)



Biological pathways: Biocarta

- Maintain "open source" pathway database, and provide lab supplies for related experiments: http://www.biocarta.com
- Pathway database is edited by "gurus".
- Very nice pathway diagrams (user created).
- No non-html output.

Biocarta pathway diagram (apoptosis)



User-curated database: Reactome

http://www.reactome.org

Gene ontology

- Gene Ontology (GO) defines a collection of words (an ontology) which are used to classify the function of a gene.
- Three broad classifications:
  - Molecular function.
  - Biological process.
  - Cellular component.
- Each of these broad terms contains a hierarchy of categories, going from general to specific.
- Each category is indexed by an identifier.

Example of GO hierarchy (apoptosis)

```
*  all : all   ( 218850 )
          o GO:0008150 : biological_process ( 145098 )
                + GO:0009987 : cellular process ( 91236 )
                      # GO:0050875 : cellular physiological process (81383 )
                            * GO:0008219 : cell death ( 2714 )
                                  o GO:0012501 : programmed cell death ( 2395 )
                                        + GO:0006915 : apoptosis ( 2061 )
                + GO:0007582 : physiological process ( 96419 )
                      # GO:0050875 : cellular physiological process ( 81383 )
                            * GO:0008219 : cell death ( 2714 )
                                  o GO:0012501 : programmed cell death ( 2395 )
                                        + GO:0006915 : apoptosis ( 2061 )
                      # GO:0016265 : death ( 3054 )
                            * GO:0008219 : cell death ( 2714 )
                                  o GO:0012501 : programmed cell death ( 2395 )
                                        + GO:0006915 : apoptosis ( 2061 )
```

Annotation available in `org.Mm.eg.db`

```
options(width=100)
ls("package:org.Mm.eg.db")
```

```
##  [1] "org.Mm.eg"                "org.Mm.eg_dbconn"          "org.Mm.eg_dbfile"
##  [4] "org.Mm.eg_dbInfo"         "org.Mm.eg_dbschema"        "org.Mm.eg.db"
##  [7] "org.Mm.egACCNUM"          "org.Mm.egACCNUM2EG"        "org.Mm.egALIAS2EG"
## [10] "org.Mm.egCHR"             "org.Mm.egCHRLENGTHS"       "org.Mm.egCHRLOC"
## [13] "org.Mm.egCHRLOCEND"       "org.Mm.egENSEMBL"          "org.Mm.egENSEMBL2EG"
## [16] "org.Mm.egENSEMBLPROT"     "org.Mm.egENSEMBLPROT2EG"   "org.Mm.egENSEMBLTRANS"
## [19] "org.Mm.egENSEMBLTRANS2EG" "org.Mm.egENZYME"           "org.Mm.egENZYME2EG"
## [22] "org.Mm.egGENENAME"        "org.Mm.egGO"               "org.Mm.egGO2ALLEGS"
## [25] "org.Mm.egGO2EG"           "org.Mm.egMAPCOUNTS"        "org.Mm.egMGI"
## [28] "org.Mm.egMGI2EG"          "org.Mm.egORGANISM"         "org.Mm.egPATH"
## [31] "org.Mm.egPATH2EG"         "org.Mm.egPFAM"             "org.Mm.egPMID"
## [34] "org.Mm.egPMID2EG"         "org.Mm.egPROSITE"          "org.Mm.egREFSEQ"
## [37] "org.Mm.egREFSEQ2EG"       "org.Mm.egSYMBOL"           "org.Mm.egSYMBOL2EG"
## [40] "org.Mm.egUNIGENE"         "org.Mm.egUNIGENE2EG"       "org.Mm.egUNIPROT"
```

GO annotation

```
na.omit( select(org.Mm.eg.db, keys = head(sigGenes),
                column = c("SYMBOL", "ENSEMBL", "GO"), keytype="ENSEMBL") )
```

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

```
##               ENSEMBL   SYMBOL         GO EVIDENCE ONTOLOGY
## 1  ENSMUSG00000079112 Fam90a1a GO:0003674       ND       MF
## 2  ENSMUSG00000079112 Fam90a1a GO:0005575       ND       CC
## 3  ENSMUSG00000079112 Fam90a1a GO:0008150       ND       BP
## 6  ENSMUSG00000020673      Tpo GO:0004447      IEA       MF
## 7  ENSMUSG00000020673      Tpo GO:0004601      IBA       MF
## 8  ENSMUSG00000020673      Tpo GO:0005509      IEA       MF
## 9  ENSMUSG00000020673      Tpo GO:0005615      IBA       CC
## 10 ENSMUSG00000020673      Tpo GO:0005615      ISO       CC
## 11 ENSMUSG00000020673      Tpo GO:0005739      IDA       CC
## 12 ENSMUSG00000020673      Tpo GO:0005886      ISO       CC
## 13 ENSMUSG00000020673      Tpo GO:0006590      IEA       BP
## 14 ENSMUSG00000020673      Tpo GO:0006979      IEA       BP
## 15 ENSMUSG00000020673      Tpo GO:0009986      ISO       CC
## 16 ENSMUSG00000020673      Tpo GO:0016020      IEA       CC
```

Detecting pathway-level changes

- Transcriptomic experiments are able to measure changes in gene expression across treatment conditions.
- Can obtain information about gene sets (e.g., GO, KEGG, Reactome).
- Allows transcriptomic data to be used to assess whether changes in expression occur at the *group* level.
- Such changes often provide greater information than single gene changes.

Over-representation analysis

- Simple approach for investigating coordinated gene expression - involves hypergeometric distribution.
- Look for functional groupings within a set of significantly differentially expressed genes:
  - e.g., what is the probability of getting 10 apoptosis genes in my 100 differentially expressed genes?
- Similar to classic hypergeometric problem:
  - e.g., what is the probability of selecting k white balls in a sample of size n from a bag containing m white and $N - m$ black balls?

Fisher's Exact Test

- In practice we can use *Fisher's Exact Test* to determine whether a functional grouping is *over-represented* (or *enriched*) in our list of differentially expressed genes.
  - This is a test for independence in a 2×2 table.
- Suppose that we observe 10 apoptosis genes in our 100 differentially expressed genes, and there are 10,000 genes in our experiment, of which 500 are apoptosis genes.
- Fisher's Exact Test uses the hypergeometric distribution to test whether being involved in apoptosis is independent of being significantly differentially expressed in our hypothetical experiment.

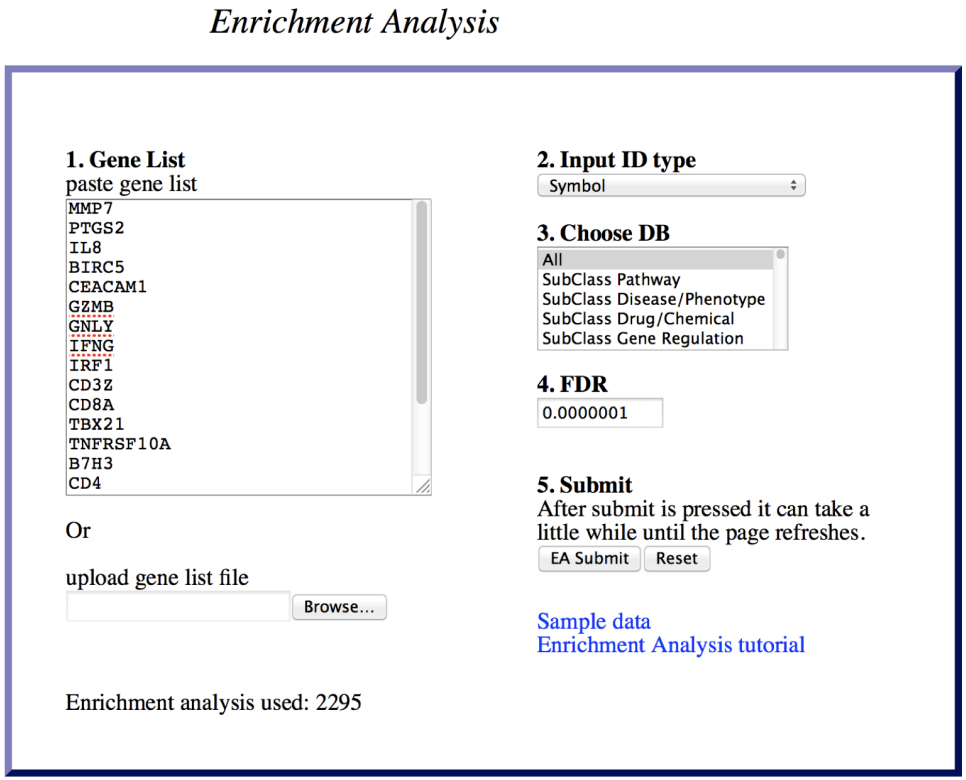
Tools for over-representation analysis

- There are MANY R-based and online tools for assessing functional enrichment of gene lists.
- We'll look at two: GeneSetDB, Enrichr
- Each provides access to multiple types of gene set annotation

Start with a list of genes

MMP7	matrix metalloproteinase 7
PTGS2	prostaglandin-endoperoxide synthase 2
IL8	interleukin 8
BIRC5	baculoviral IAP repeat-containing 5
CEACAM1	carcinoembryonic antigen-related cell adhesion molecule 1
GZMB	granzyme B
GNLY	granulysin
IFNG	interferon, gamma
IRF1	interferon regulatory factor 1
CD3Z	CD3Z antigen, zeta polypeptide
CD8A	CD8 antigen, alpha polypeptide
TBX21	T-box 21
TNFRSF10A	tumor necrosis factor receptor superfamily, member 10a
B7H3	B7 homolog 3
CD4	CD4 antigen (p55)
IL10	interleukin 10
TGFB1	transforming growth factor, beta 1
VEGF	vascular endothelial growth factor

GeneSetDB: input



http://genesetdb.auckland.ac.nz

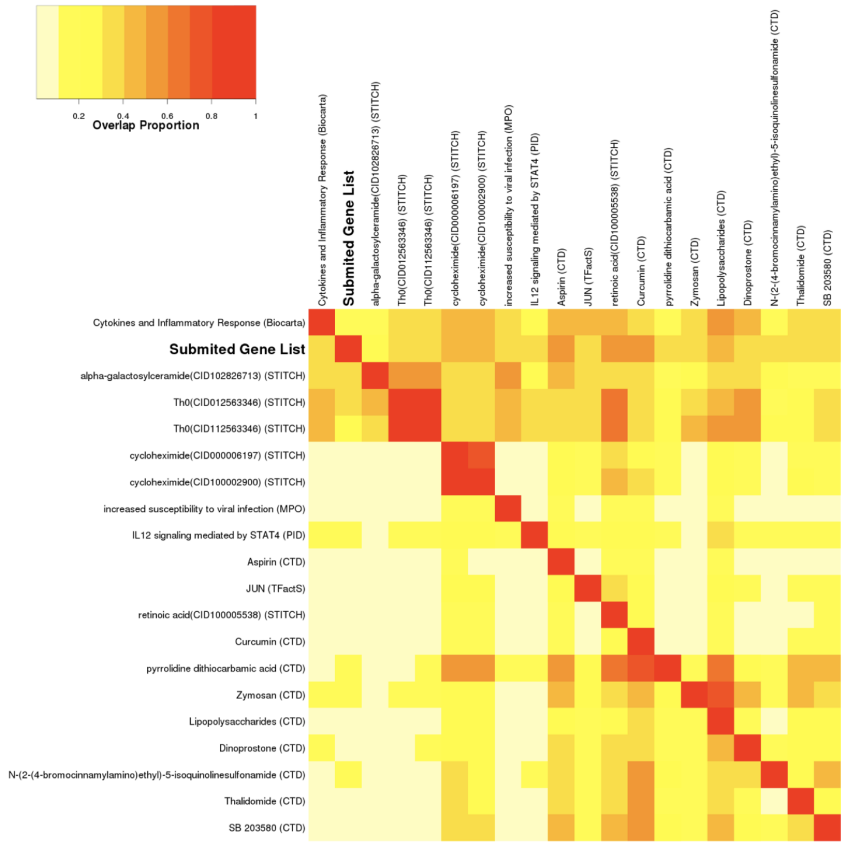
GeneSetDB: output

18 symbol input ids were converted into 15 unique gene ids.
19 entries for fdr cutoff 0.0000001 estimated.

Sub Class	Gene Set Name	Source DB	Gene #	Gene # with Anno	Gene # without Anno	p-value	FDR
Drug/Chemical	Th0(CID012563346)	STITCH	16	5	11	6.9E-13	2.2E-9
Drug/Chemical	Th0(CID112563346)	STITCH	18	5	13	1.4E-12	2.2E-9
Pathway	Cytokines and Inflammatory Response	Biocarta	26	5	21	1.0E-11	1.1E-8
Drug/Chemical	Aspirin	CTD	461	9	452	1.4E-11	1.1E-8
Drug/Chemical	Dinoprostone	CTD	90	6	84	5.1E-11	2.3E-8
Drug/Chemical	cycloheximide(CID100002900)	STITCH	179	7	172	4.0E-11	2.3E-8
Pathway	IL12 signaling mediated by STAT4	PID	35	5	30	5.1E-11	2.3E-8
Disease/Phenotype	increased susceptibility to viral infection	MPO	93	6	87	6.3E-11	2.5E-8
Drug/Chemical	Zymosan	CTD	39	5	34	9.0E-11	3.2E-8
Drug/Chemical	Lipopolysaccharides	CTD	204	7	197	1.0E-10	3.2E-8
Drug/Chemical	N-(2-(4-bromocinnamylamino)ethyl)-5-isoquinolinesulfonamide	CTD	42	5	37	1.3E-10	3.9E-8
Drug/Chemical	SB 203580	CTD	108	6	102	1.6E-10	4.2E-8
Drug/Chemical	alpha-galactosylceramide(CID102826713)	STITCH	13	4	9	1.9E-10	4.7E-8
Drug/Chemical	pyrrolidine dithiocarbamic acid	CTD	46	5	41	2.1E-10	4.9E-8
Drug/Chemical	cycloheximide(CID000006197)	STITCH	233	7	226	2.6E-10	5.4E-8
Drug/Chemical	Thalidomide	CTD	125	6	119	3.8E-10	6.8E-8
Drug/Chemical	retinoic acid(CID100005538)	STITCH	421	8	413	3.5E-10	6.8E-8
GeneRegulation	JUN	TFactS	125	6	119	3.8E-10	6.8E-8
Drug/Chemical	Curcumin	CTD	438	8	430	4.8E-10	8.1E-8

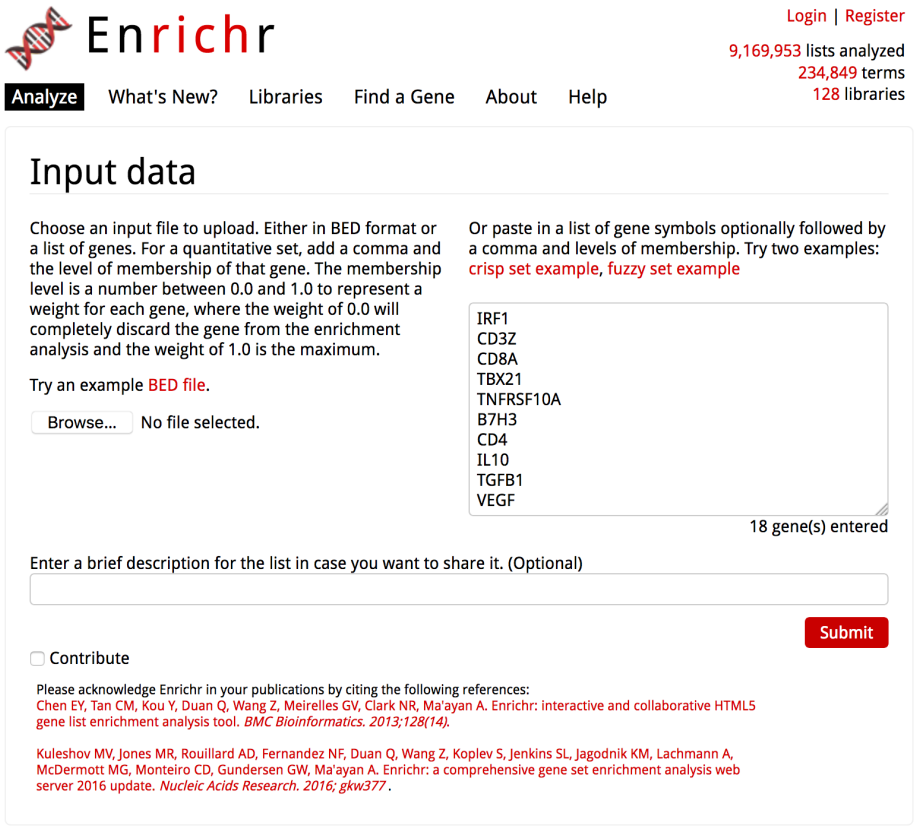
http://genesetdb.auckland.ac.nz

GeneSetDB: gene set overlap



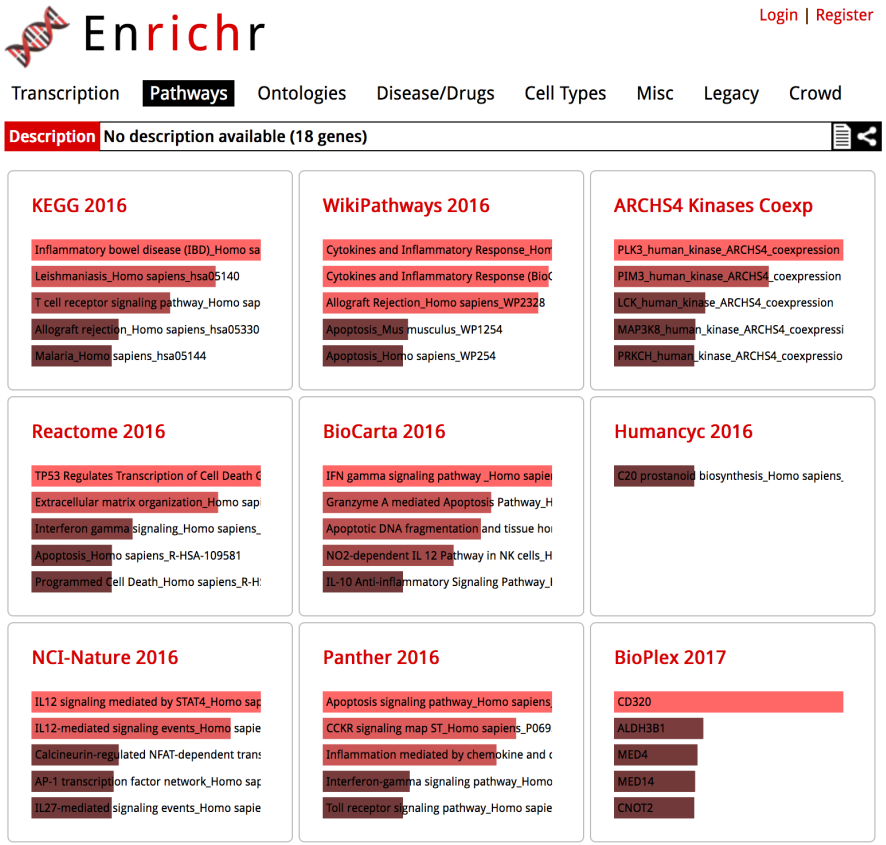
http://genesetdb.auckland.ac.nz

Enrichr



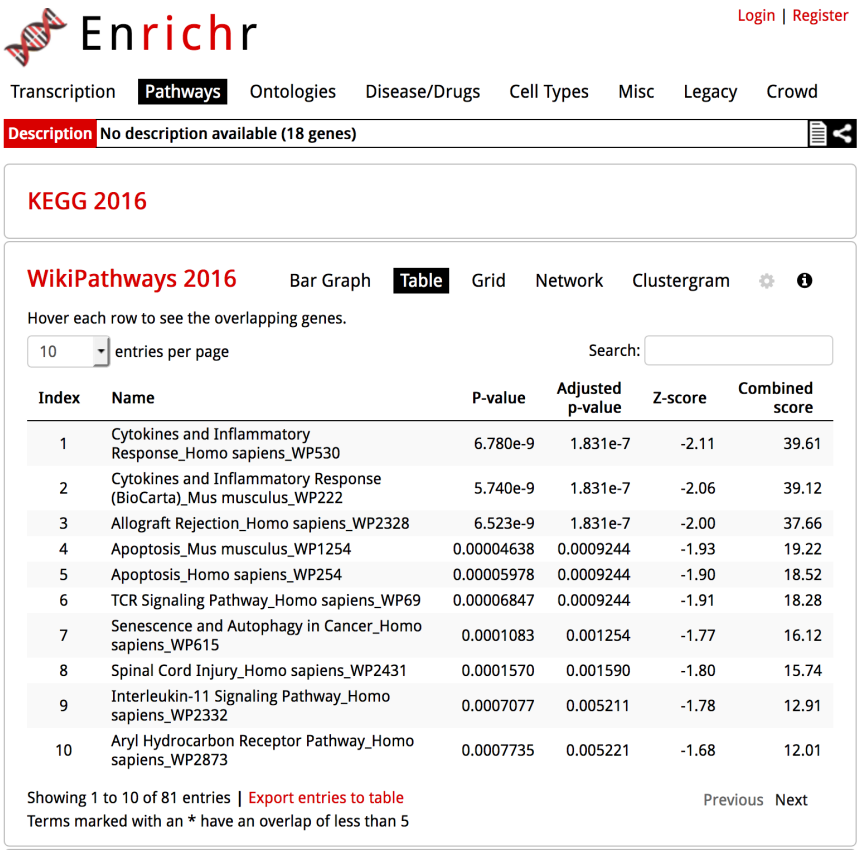
http://amp.pharm.mssm.edu/Enrichr/

Enrichr



http://amp.pharm.mssm.edu/Enrichr/

Enrichr



Index	Name	P-value	Adjusted p-value	Z-score	Combined score
1	Cytokines and Inflammatory Response_Homo sapiens_WP530	6.780e-9	1.831e-7	-2.11	39.61
2	Cytokines and Inflammatory Response (BioCarta)_Mus musculus_WP222	5.740e-9	1.831e-7	-2.06	39.12
3	Allograft Rejection_Homo sapiens_WP2328	6.523e-9	1.831e-7	-2.00	37.66
4	Apoptosis_Mus musculus_WP1254	0.00004638	0.0009244	-1.93	19.22
5	Apoptosis_Homo sapiens_WP254	0.00005978	0.0009244	-1.90	18.52
6	TCR Signaling Pathway_Homo sapiens_WP69	0.00006847	0.0009244	-1.91	18.28
7	Senescence and Autophagy in Cancer_Homo sapiens_WP615	0.0001083	0.001254	-1.77	16.12
8	Spinal Cord Injury_Homo sapiens_WP2431	0.0001570	0.001590	-1.80	15.74
9	Interleukin-11 Signaling Pathway_Homo sapiens_WP2332	0.0007077	0.005211	-1.78	12.91
10	Aryl Hydrocarbon Receptor Pathway_Homo sapiens_WP2873	0.0007735	0.005221	-1.68	12.01

http://amp.pharm.mssm.edu/Enrichr/

Limitations of enrichment testing

- The hypergeometric-based enrichment tests only take the size of gene sets into account.
- All genes for the same group that are not significant are treated the same.
  - What if they are "almost" significant?
  - We are now thinking about the *ranks* of the genes.
  - Can we incorporate this rank information into our calculations?
- More sosphiticated methods (e.g., Gene Set Enrichment Analysis) take a rank-based approach.

We can stop here...

Detecting differential expression: DESeq2

- The DESeq2 package uses the *Negative Binomial* distribution to model the count data from each sample.
- Statistical test based on the Negative Binomial distribution (via a generalized linear model, GLM) can be used to assess differential expression for each gene.
- Negative Binomial distribution attempts to accurately capture the variation that is observed for count data.

M. I. Love, W. Huber, S. Anders: Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology 2014, 15:550.

Detecting differential expression: DESeq2

```
library(DESeq2)
# Specify "conditions" (groups: WT and KO)
# Create object of class CountDataSet derived from eSet class
dds <- DESeqDataSetFromMatrix(countData = counts,
                              colData = data.frame(groups),
                              design = ~groups)
counts(dds)[1:4, ]
```

```
FALSE                     WT1 WT2 WT3 KO1 KO2 KO3
FALSE ENSMUSG00000000702   0   0   0   0   0   0
FALSE ENSMUSG00000078423   0   0   0   0   0   0
FALSE ENSMUSG00000078424   0   0   0   0   0   0
FALSE ENSMUSG00000071964   0   0   0   0   0   0
```

Detecting differential expression: DESeq2

```
## Fit DESeq model to identify DE transcripts
dds <- DESeq(dds)
res <- DESeq2::results(dds)
## Remove rows with NAs
res = na.omit(res)
options(digits=2, width=100)
head(res, 4)
```

```
## log2 fold change (MLE): groups WT vs KO
## Wald test p-value: groups WT vs KO
## DataFrame with 4 rows and 6 columns
##                    baseMean log2FoldChange     lfcSE      stat    pvalue      padj
##                   <numeric>      <numeric> <numeric> <numeric> <numeric> <numeric>
## ENSMUSG00000063889        69          -0.35      0.28      -1.2    0.2131     0.454
## ENSMUSG00000024231       414          -0.70      0.21      -3.3    0.0011     0.015
## ENSMUSG00000024232       284           0.53      0.17       3.1    0.0018     0.023
## ENSMUSG00000073647        99          -0.76      0.35      -2.2    0.0312     0.145
```

DESeq2: adjusted p-values

```
sum(res$padj <= 0.05)
```

```
## [1] 2043
```

```
## Get the rows of "res" with significant adjusted p-values
resPadj<-res[res$padj <= 0.05 , ]
```

Detecting differential expression: edgeR

- The edgeR package also uses the negative binomial distribution to model the RNA-seq count data.
- Takes an empirical Bayes approach to the statistical analysis, in a similar way to how the `limma` package handles transcriptomic data.

Detecting differential expression: edgeR

```
library(edgeR)
# Construct DGEList object
y <- DGEList(counts=counts, group=groups)

# Calculate library size (counts per sample)
y <- calcNormFactors(y)

# Estimate common dispersion (overall variability)
y <- estimateCommonDisp(y)

# Estimate tagwise dispersion (per gene variability)
y <- estimateTagwiseDisp(y)

# Compute exact test for the negative binomial distribution.
et <- exactTest(y)
```

Detecting differential expression: edgeR

```
topTags(et, n=4)$table
```

```
##                     logFC logCPM   PValue     FDR
## ENSMUSG00000070034  -7.6    8.7 1.8e-103 6.9e-99
## ENSMUSG00000079455  -9.6    7.5  2.1e-98 3.9e-94
## ENSMUSG00000033634  -8.8    6.5  1.2e-97 1.5e-93
## ENSMUSG00000024552  -8.4    4.8  1.2e-92 1.1e-88
```

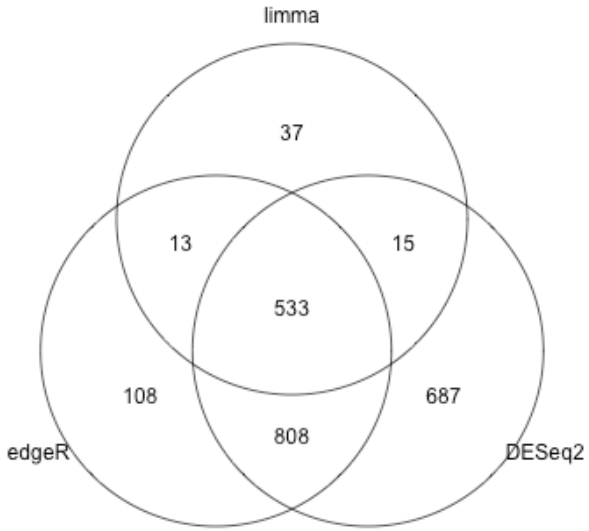
```
edge <- as.data.frame(topTags(et, n=nrow(counts)))
sum(p.adjust(edge$FDR <= 0.05))
```

```
## [1] 1462
```

```
## Get the rows of "edge" with significant adjusted p-values
edgePadj <- edge[edge$FDR <= 0.05, ]
```

limma vs edgeR vs DESeq2

```
library(gplots)
venn(list(edgeR=rownames(edgePadj), DESeq2=rownames(resPadj), limma=rownames(limmaPadj)))
```



Other normalisation methods (limma-voom)

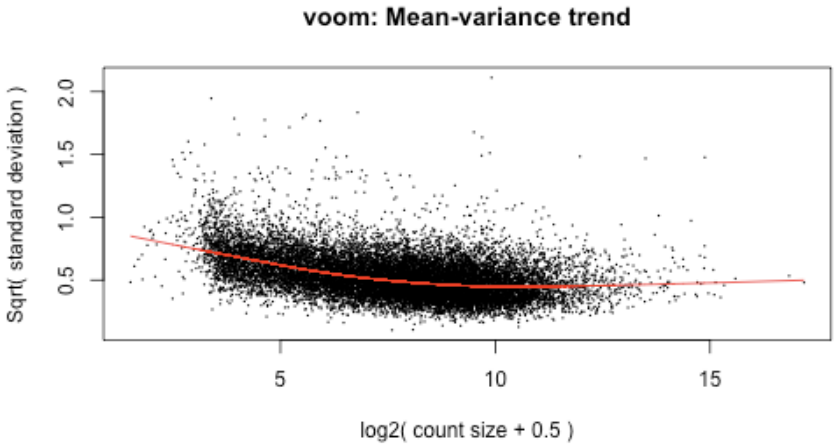
Re-make our original DGEList object:

```
## Create DGEList object from count data
dge <- DGEList(counts=counts)

## Figure out which genes to keep
keep <- filterByExpr(dge, design)

## Apply filtering and recalculate library sizes
dge <- dge[keep, keep.lib.sizes=FALSE]

## Calculate normalisation factor (i.e., account for total reads per sample)
dge <- calcNormFactors(dge)
```

limma-voom

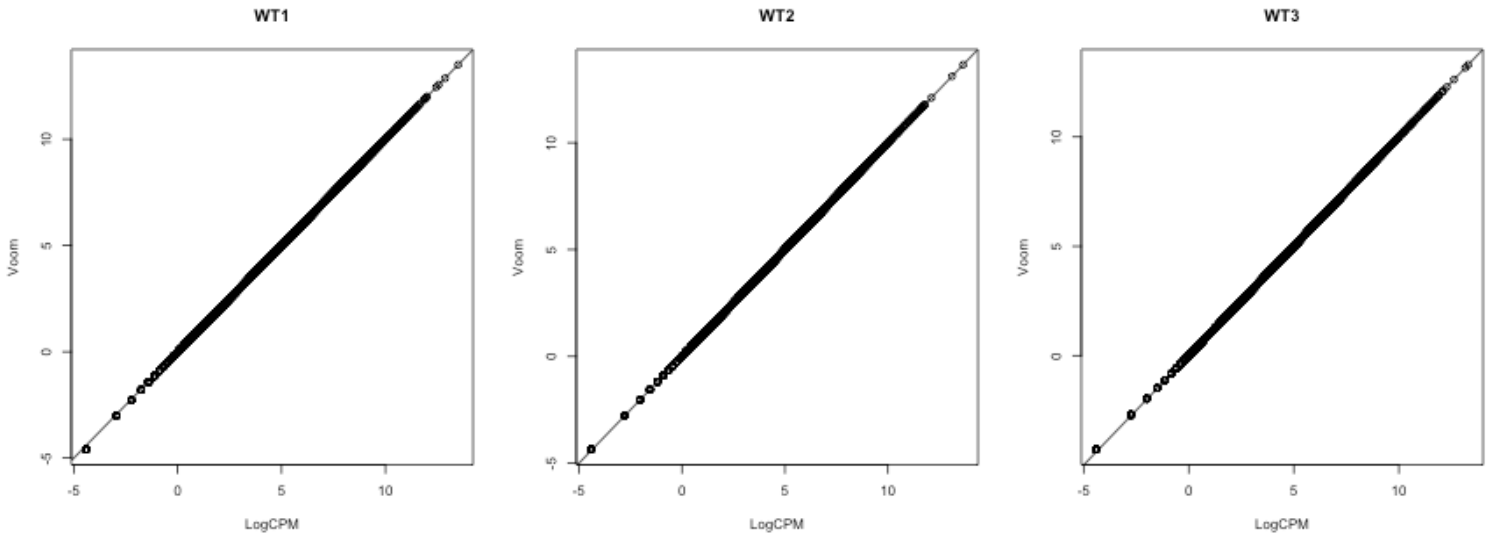
- The "voom" function estimates relationship between the mean and the variance of the logCPM data, normalises the data, and creates "precision weights" for each observation that are incorporated into the limma analysis.

```
v <- voom(dge, design, plot=TRUE)
```



Limma-voom (impact on first three samples)

```
par(mfrow=c(1,3))
for(i in 1:3){
  plot(logCPM[,i], v$E[,i], xlab="LogCPM", ylab="Voom", main=colnames(logCPM)[i])
  abline(0,1)
}
```



limma-voom

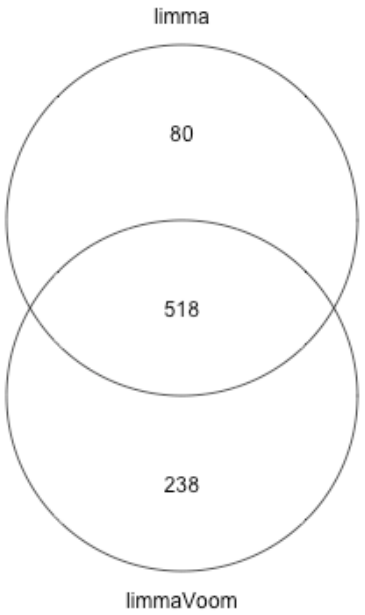
```
options(digits=4)
fit <- lmFit(v, design)
fit <- eBayes(fit)
tt <- topTable(fit, coef=ncol(design), n=nrow(v))
head(tt)
```

```
##                     logFC AveExpr      t   P.Value adj.P.Val      B
## ENSMUSG00000025815 -4.265  4.8430 -24.37 5.556e-09 4.539e-05 11.251
## ENSMUSG00000070034 -7.565  5.8920 -24.00 6.287e-09 4.539e-05 11.013
## ENSMUSG00000064201  4.491  8.9488  21.30 1.662e-08 4.539e-05 10.413
## ENSMUSG00000033191 -3.586  6.9315 -21.29 1.667e-08 4.539e-05 10.411
## ENSMUSG00000032204 -2.921  6.0643 -18.20 5.939e-08 8.432e-05  9.170
## ENSMUSG00000086445 -7.511 -0.6625 -22.31 1.139e-08 4.539e-05  8.831
```

```
## Get the rows of top table with significant adjusted p-values
limmaVoomPadj <- tt[tt$adj.P.Val <= 0.05, ]
```

limma vs limma-voom

```
venn( list(limmaVoom = rownames(limmaVoomPadj), limma = rownames(limmaPadj)) )
```



limma-voom vs edgeR vs DESeq2

```
venn(list(edgeR = rownames(edgePadj), DESeq2 = rownames(resPadj),
          limmaVoom = rownames(limmaVoomPadj)))
```