**DESeq2:**

**Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2**. Michale Love, W. Huber, and Simon Anders. Genome Biology. 2014.

A core challenge of in typical HTS experiments is the small number of samples in a given experiment. Inferential methods that treat each gene separately often suffer from a lack of power, due to the high uncertainty of within-group varianc­e estimates. In high throughput assays, this limitation can be overomce by pooling information across gene, specifically, by exploiting assumptions about the similarity of the variances of different genes measure in the same experiment.

Many methods for differential expression analysis of RNAseq data peform such information sharing across genes for variance estimation.

The most common approach to the comparative analysis of transcriptomics data is to test the null hypothesis that the logarithmic fold change (LFC) between treated and control for a gene’s expression is exactly zero (i.e. that the gene is not affected by the treatment). Often the goal of differential analsysis is to produce a list of gene passing multiple-test adjustment, ranked by P value. However, small changes, even if statistically significant, mihght not be the most interesting candidates for further investigation. Instead, ranking by LFC on the other hand is complicated by the noisiness of LFC estimates for genes with low counts.

In this paper, the authors present DESeq2 package, which is a successor to their DESeq method. DeSeq2 intesgrates methodological advances with several novel features to facilitate a more quantitative analsysis of comparative RNA-seq data using shrinkage estimators for dispersion and fold change.

In this method, they fit a generalized linear model (GLM) for each gene/transcript, using a negative binomial distribution.

When you have few samples, you tend to have high dispersion/variance of the counts across your replicates. If used directly, these noisy estimates would compromise the accuracy of differential expression testing. To deal with this, in DESeq2, they share information across genes, assuming that genes of similar average expression strength have similar dispersion.

1. Treat each gene separately, and estimate gene-wise dispersion estimates (using maximum likelihood).
2. Determine the location parameter of the distribution of these estimates; to allow for dependence on average expression strength, they fit a smooth curve. This provides an accurate estimate for the expected diserpsion value for genes of a given expression strength.
3. Shrink the gene-wise dispersion estimates toward the values predicted by the curve to obtain final dispersion values.

The DESeq2 approach is similar to the one used by DSS, in that both methods sequentially estimate a prior distribution for the true dispersion values around the fit, and then provide the maximum posteriori (MAP) as the final estimate.

A common difficulty in the analysis of HTS data is the strong variance of LFC estimates for genes with low read counts. In other words, genes with low expression tend to show much stronger differences than more highly expressed genes. This is known as heteroskedasticity, in which the variance of LFCs depend on the mean count. This complicates downstream analyses.

**Hypothesis testing for differential expression.**

After GLMs are fit for each gene, DESeq2 uses a Wald test to do significance testing. The shrunken estimate of LFC is dividedby its standard error, resulting in a z-statistic, which is compared to a standard normal distribution.

The Wald test P-values from the subset of genes that pass an independent filtering step are adjusted for multiple testing using the procedure of Benjamini and Hochberg (adjusted p-value). This is typically associated with a loss of power. To minimize the number of true differentially expressed genes from being removed, DESeq2 omits genes that have little to no chance of being detected as differentially expressed. This step relies on the average expression strength of each gene, across all sample, as its filter criterion, and it omits genes with mean normalized counts below a filtering threshold from multiple testing adjustment. By default, DESeq2 will choose a threshold that maximizes the number of genes found at a user-specified FDR.

**Regularized log transformation**

Useful for transforming data such that it is homoskedastic (i.e. the variance stays similar across gene counts).

With raw counts, the results will be dominated by highly expressed genes. In log transformed data, undue weight will be given to weakly expressed genes, which show exaggerated LFCs. This is useful for visualization (e.g. PCA, clustering) but is not a part of the statistical inference procedure for differential expression analysis.

**DESeq2 is hosted on Bioconductor and is used in R**

*http://bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html*

**Statquest: DESeq2, part 1, library normalization**

* RPKM, FPKM, TPM deal with normalization problems due to sequencing depth.
* The normalization can handle

1. Differences in library sizes
2. differences in library composition

* Calculate a scaling factor for each sample – takes into account the library composition and the read depth into account.
* **DESeq2:** is a Bioconductor package that is used to take gene or transcript count data (in table form) for a number of different samples, and normalizes the data and provides functions that allow for the quantification and statistical inferences of systematic changes between conditions. The package provides methods to test for differential expression by use of negative binomial generalized linear models; the estimates of dispersion and logarithmic fold changes incorporate data-driven prior distributions.
* *Step 1:* first takes the log of all of the reads for each sample (log base e), so these numbers are what we would need to raise e to, in order to get the original value.
* DEseq2 uses log*e* since that is the default in R.
* *Step 2: take averages across your samples:* log*e* 0 = -inf; therefore if you average across the samples, you’ll get an average of infinity or –infinity. Using log, you are less likely to be swayed by outliers (geometric averages).
* *Step 3: Filter out genes that have averages of infinity.*
* In general, this step filters out genes with zero read counts in one or more samples. In theory, this helps focus the scaling factors on the house keeping genes – genes transcribed at similar levels regardless of tissue type.

*Step 4: Subtract the average log value from the log(counts)*

*For each gene in each sample, subtract the average for that gene across all samples.*

Log(reads for gene X) – log(average for gene X) = log(reads for gene X/ average for gene X)

*Step 5: Calculate the median of the ratios for each sample.*

For each sample, identify the median of the ratios of all genes.

Using the median is another way to avoid extreme genes from swaying the value too much in one direction

*Step 6: Convert the medians to “normal numbers” to get the final scaling factors for each sample*

At this point, the medians are log values, so they are exponents (in this case, exponents for e)

Raise e to the median of each sample to get the scaling factor (e^median)

*Step 7: Divide the original read counts by the scaling factors.*

*Summary of DEseq2’s library size scaling factor:*

Logs eliminate all genes that are only transcribed in one sample type (liver vs. spleen). They also help smooth over outlier read counts (via geometric mean).

The median further downplays genes that soak up a lot of the reads, putting more emphasis on moderately expressed genes.

**StatQuest: edgeR and DESeq2, part 2 - Independent Filtering**

Filtering out genes with low read counts aka independent filtering

Say you rely on a p-value of 0.01; if you sequence 20,000 genes in an RNAseq experiment, you would get 1000 false positives!

FDR and Benjamini-Hochberg correct for this.

Watch Statquest on FDR!

FDR-adjustment, is used to correct for multi-gene testing. Simple fact of the testing thousands of different gene expressions. Even if you have a p<0.05, you have a chance of getting many false positives if you sample enough tests. By using an FDR cutoff (usually <0.05), you reduce the number of true positives, but you also greatly reduce the false positives.

It is in our interest to filter out bogus tests, since the more you have, the more true positives are lost.

edgeR suggests keeping only genes that >1 cpm in 2 or more samples

CPM = counts per million

Compensates for differences in read depth in libraries

Take reads for Gene X in sample Y and divide by total reads for sample Y divided by 1x10^6 (divide by 1x10^6, since your numbers would be tiny)

Should try different cpm cutoffs. Should calculate p-values before trying different cutoffs for the minimum CPM.

Difference between edgeR and DEseq2:

EdgeR – keeps genes with > min. CPM in 2 or more samples

DESeq2 keeps genes with average CPM> min CPM, plots significant genes vs. quantiles, fits curve, and takes a max-noise = threshold.

DEseq2’s method can be applied to find the optimal CPM to edgeR’s gene selection criteria.

**DataCamp - Course: RNA-seq Differential Expression Analysis:**

We need an appropriate statistical model

We will be using the negative binomial model, which can accurately model the count data

Preparation for differential expression analysis:

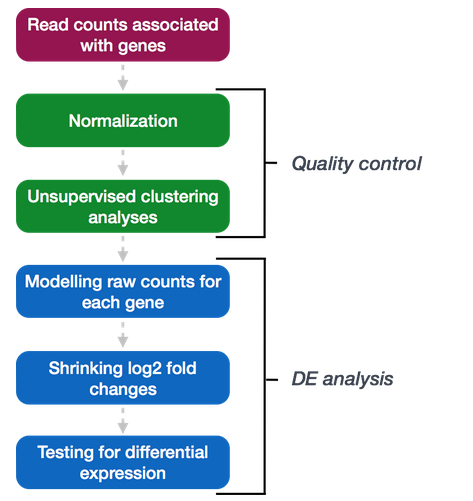
rawcounts, sample metadata

Differential expression analysis:

DESeq2, edgeR, and limma/voom

limma/voom may not be sensitive enough for experiments that don’t have very many biological replicates.

Steps in differential expression analysis:



Why do we use ***unnormalized*** counts as input for DESeq2? DESeq2 assumes the input is raw counts and will internally correct for the library size when performing the analysis.

organizing your data: the row names of the metadata needs to be in the same order of the colnames of your raw counts data.

This can be checked with:

all(rownames(wt\_metadata) == colnames(wt\_rawcounts)) [1] FALSE

Matching order between vectors:

idx <- match(colnames(wt\_rawcounts), rownames(wt\_metadata))

reordered\_wt\_metadata <- wt\_metadata[idx, ]

all(rownames(reordered\_wt\_metadata) == colnames(wt\_rawcounts))

[1] TRUE

Creating the DeSeq2 object:

dds\_wt <- DESeqDataSetFromMatrix(countData = wt\_rawcounts, colData = reordered\_wt\_metadata, design = ~ condition)

**Count normalization:**

need to normalize for Library size, gene length, base composition.

Gene length only needs to be normalized if you want to compare expression across different genes within the same sample.

To calculate normalized counts:

dds\_wt <- estimateSizeFactors(dds\_wt)

sizeFactors(dds\_wt)

deseq2 will use these sizefactors to normalize the raw counts

Normalized counts can then be extracted using the counts function.

# extract the normalized counts:

normalized\_wt\_counts <- counts(dds\_wt, normalized=TRUE)

# extract the raw counts:

normalized\_wt\_counts <- counts(dds\_wt, normalized=FALSE)

**Unsupervised clustering analysis (QC): hierarchical heatmap:**

# Transform across the normalized counts:

vsd\_wt <- vst(dds\_wt, blind=TRUE)

# To set up the correlation heatmap:

# Extract the vst matrix from the object

vsd\_mat\_wt <- assay(vsd\_wt)

# Compute pairwise correlation values

vsd\_cor\_wt <- cor(vsd\_mat\_wt) View(vsd\_cor\_wt)

# Then use pheatmap:

library(pheatmap)

# Plot heatmap

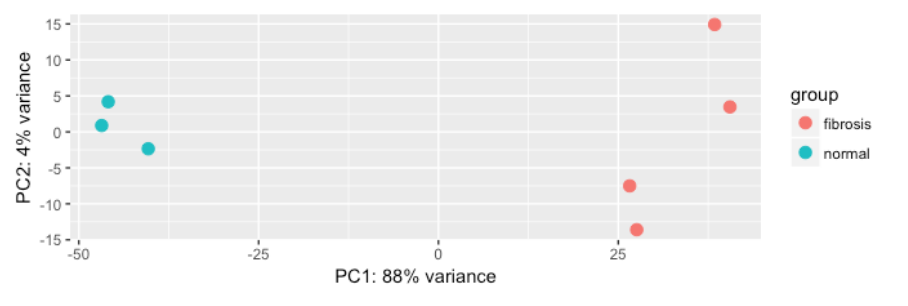
pheatmap(vsd\_cor\_wt, annotation = select(wt\_metadata, condition))

This is useful for identifying outliers

**Principal Component Analysis (PCA)**

# Plot PCA (color samples according to condition)

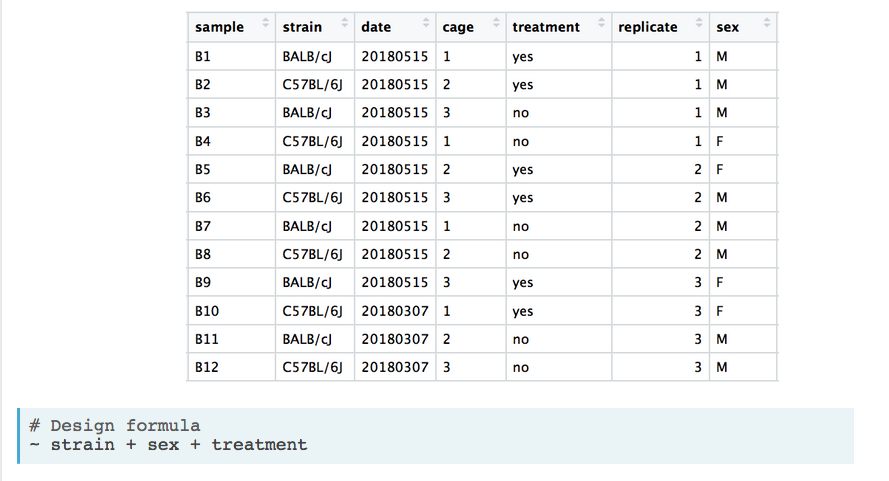
plotPCA(vsd\_wt, intgroup="condition")



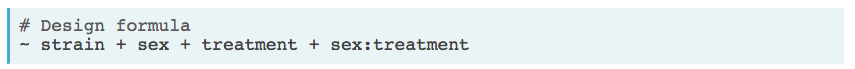
You want to see your groups/conditions separated by PC1. If not, there may be other sources of variation.

**Overview of the DE Analysis:**

After QC, we may need to re-create the dds object if we found additional sources of variation or if we removed outliers.



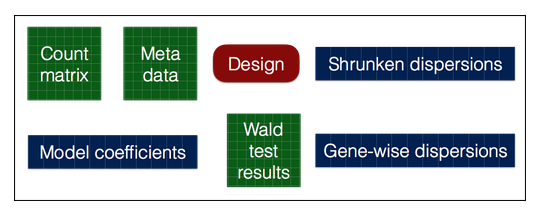
You can incorporate an interaction (for example seeing how sex affects the treatment effect) into your model by doing the following:



Once you have your design you can run your DEseq2 analysis:

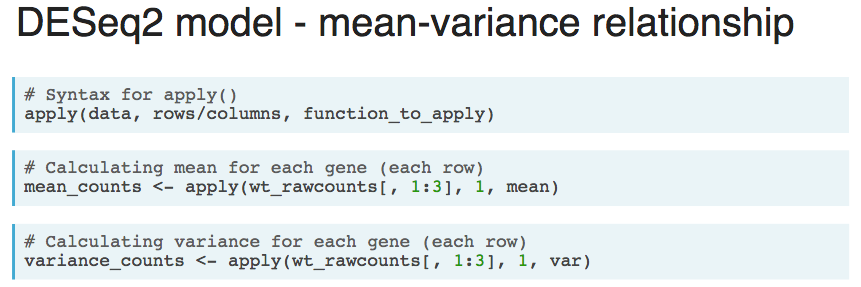
# Run analysis

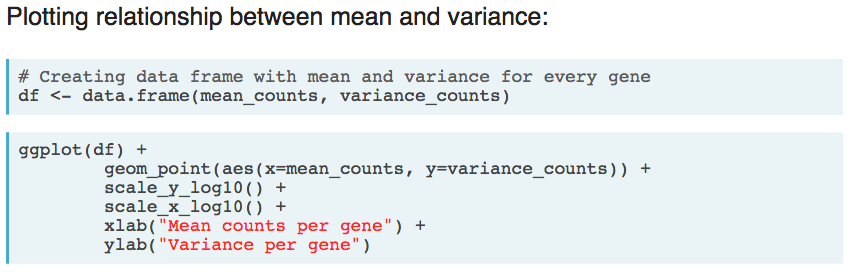
dds\_wt <- DESeq(dds\_wt)



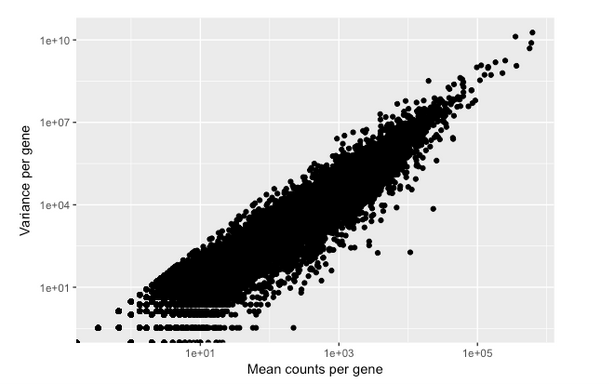
DESeq2 model:

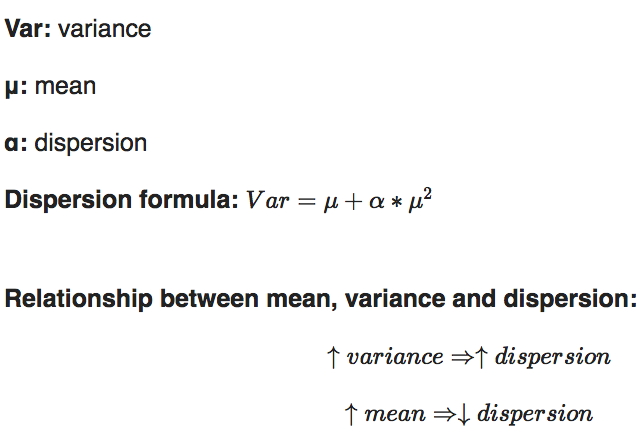
How well does our data fit the model?

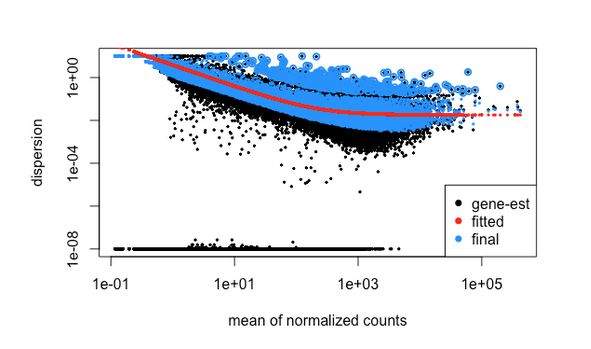
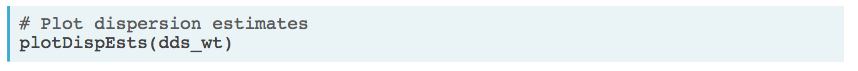


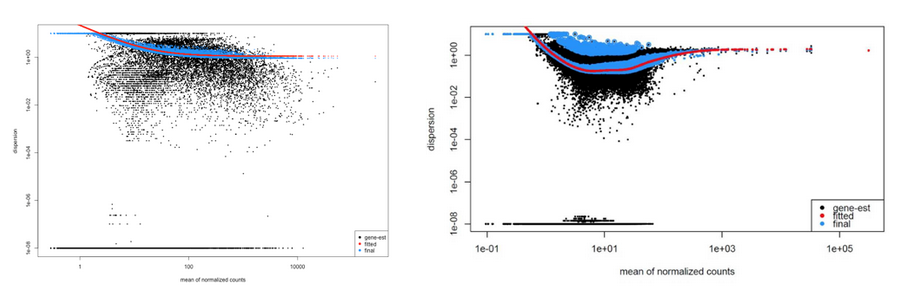
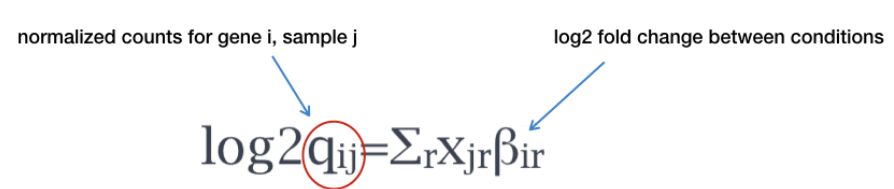
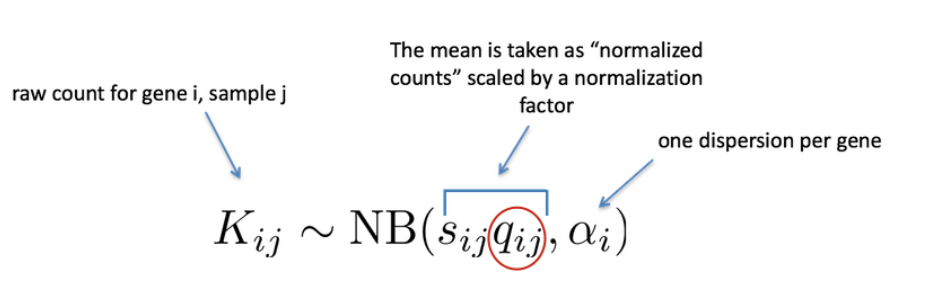


Variance per gene increases as the mean counts per gene increases.







Worrisome plots (explained by outliers or sample outliers): 

By default DESeq2 will use the Wald test for pair wise comparisons to test for differences in expression for the 2 conditions of interest.

Results can be extracted by:

results(wt\_dds, alpha = 0.05)

# lower alpha is more strict; 0.05 is standard

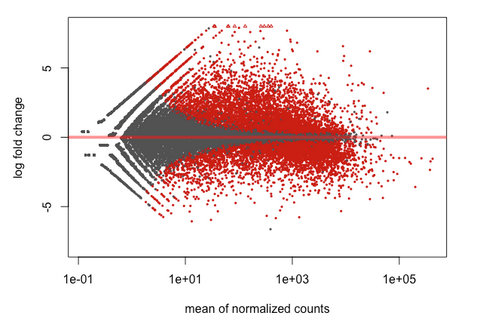
Set up the contrasts:

results(dds, contrast = c("condition\_factor", "level\_to\_compare", "base\_level"), alpha = 0.05)

wt\_res <- results(dds\_wt, contrast = c("condition", "fibrosis", "normal"), alpha = 0.05)

The MA plot plots the mean of normalized counts against log2 fold change.

plotMA(wt\_res, ylim=c(-8,8))

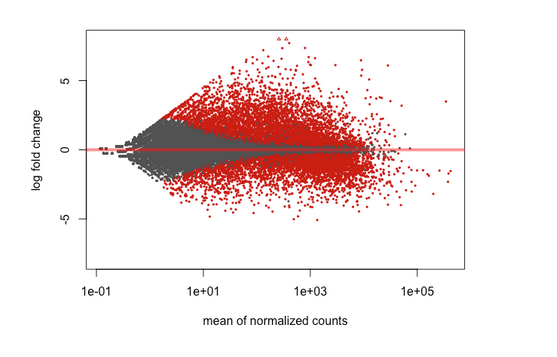


LFC shrinkage:

wt\_res <- lfcShrink(dds\_wt, contrast=c("condition", "fibrosis", "normal"), res=wt\_res)

# Then regenerate the MA plot:

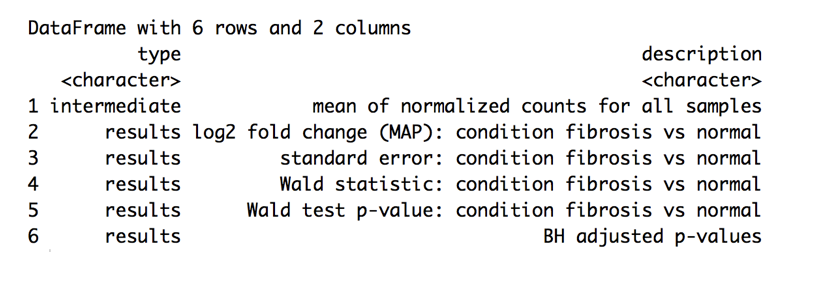
plotMA(wt\_res, ylim=c(-8,8))



**DESeq2 results:**

To view the results table:

mcols(wt\_res)



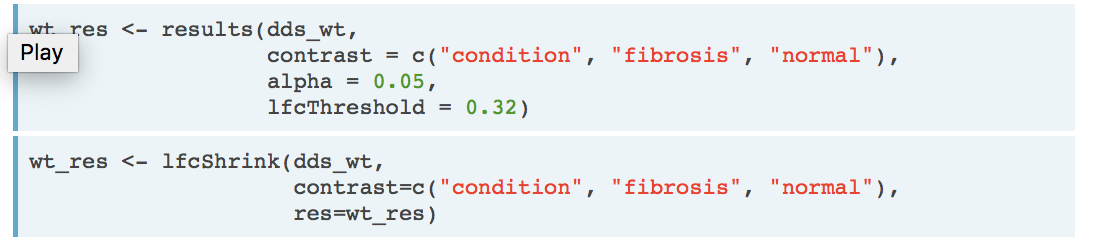
At alpha = 0.05, expect 5% of your genes will be detected as significant (false positives). Out of 47000 genes, that yields about 2300 false positives.

With Benjamini Hochberg correction (adjusted p value): If you have 1000 differentially expressed genes, 5% would be false positives (50 genes)

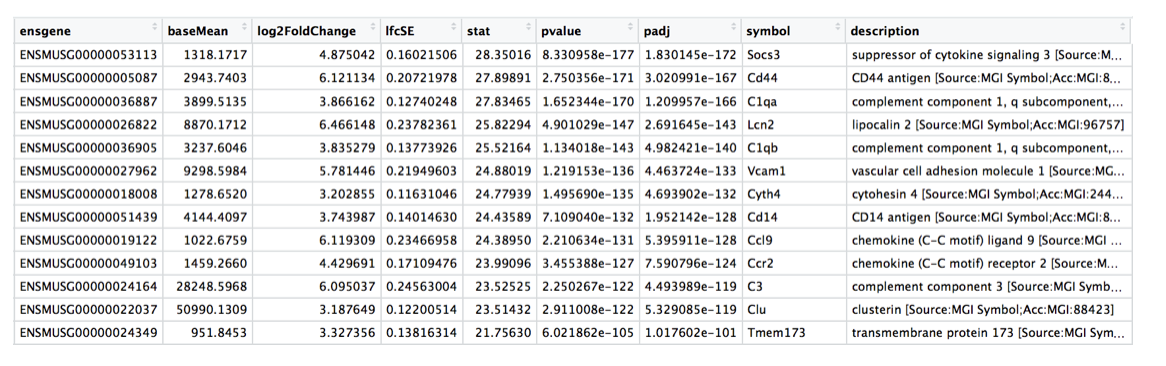
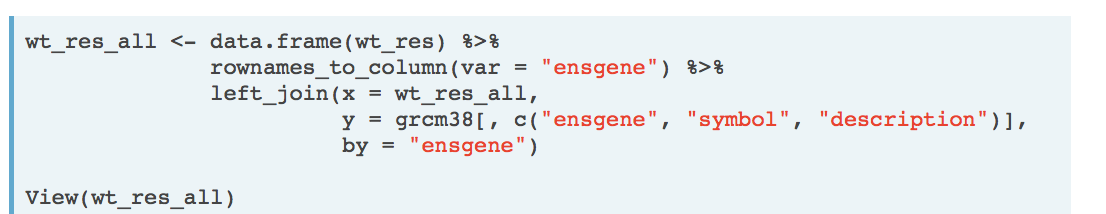
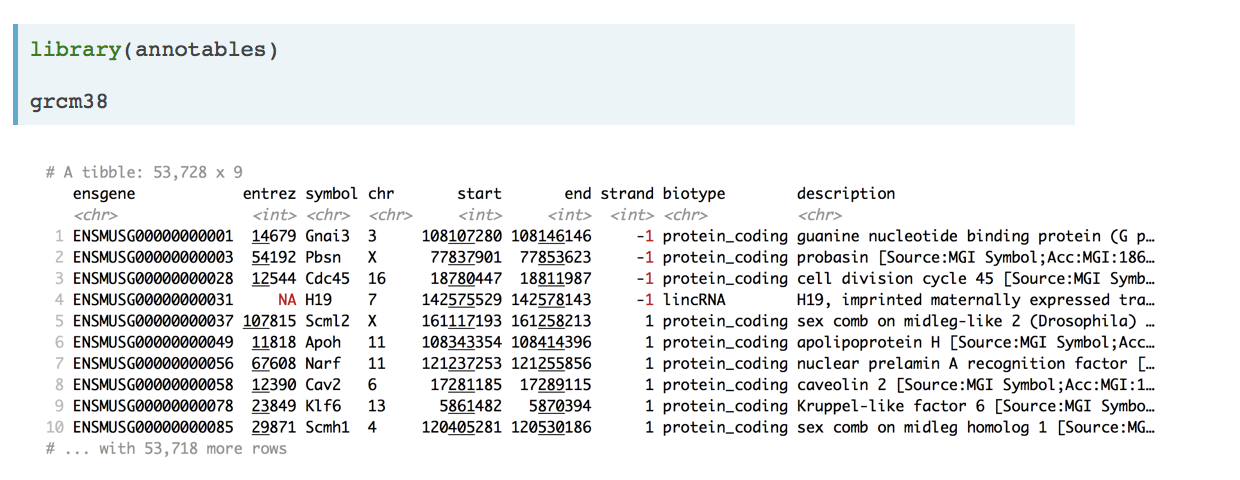
DESseq2 automatically removes genes that are unlikely to be significantly different (minimize the multiple testing problem) – example: genes that have 0 counts across samples, low mean count values, extreme count outliers

summary(wt\_res)

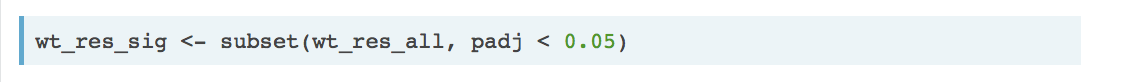
Sometimes you may have a large number of DEGs. Common to choose a fold change cut-off.

To choose a fold change threshold

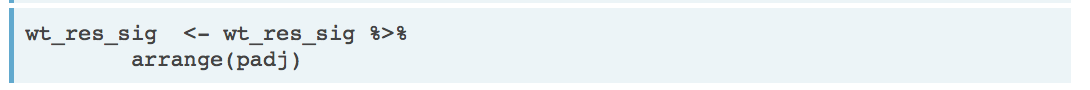
Annotate your gene list, use the annotables package:



Subset the significant genes using the subset function:



Then sort according to padj:



Visualizing the results:

Heatmap:

# Subset the significant genes:

sig\_norm\_counts\_wt <- normalized\_counts\_wt[wt\_res\_sig$ensgene, ]

# Choose a color palette from Rcolorbrewer:

library(RColorBrewer)

heat\_colors <- brewer.pal(6, "YlOrRd")

# Run pheatmap – scale by row (z=scores)

pheatmap(sig\_norm\_counts\_wt,

color = heat\_colors,

cluster\_rows = T,

show\_rownames = F,

annotation = select(wt\_metadata, condition),

scale = "row")

# Volcano plot:

# Obtain logical vector regarding whether padj values are less than 0.05

wt\_res\_all <- wt\_res\_all %>%

rownames\_to\_column(var = "ensgene") %>%

mutate(threshold = padj < 0.05)

# Volcano plot

ggplot(wt\_res\_all) +

geom\_point(aes(x = log2FoldChange, y = -log10(padj),

color = threshold)) +

xlab("log2 fold change") +

ylab("-log10 adjusted p-value") +

theme(legend.position = "none",

plot.title = element\_text(size = rel(1.5), hjust = 0.5),

axis.title = element\_text(size = rel(1.25)))

# Expression plot (top 20):

top\_20 <- data.frame(sig\_norm\_counts\_wt)[1:20, ] %>%

rownames\_to\_column(var = "ensgene")

top\_20 <- gather(top\_20,

key = "samplename",

value = "normalized\_counts",

2:8)

top\_20 <- inner\_join(top\_20,

rownames\_to\_column(wt\_metadata, var = "samplename"),

by = "samplename")

ggplot(top\_20) +

geom\_point(aes(x = ensgene, y = normalized\_counts, color = condition)) +

scale\_y\_log10() +

xlab("Genes") +

ylab("Normalized Counts") +

ggtitle("Top 20 Significant DE Genes") +

theme\_bw() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +

theme(plot.title = element\_text(hjust = 0.5)

**RNA-seq DE Analysis Summary:**