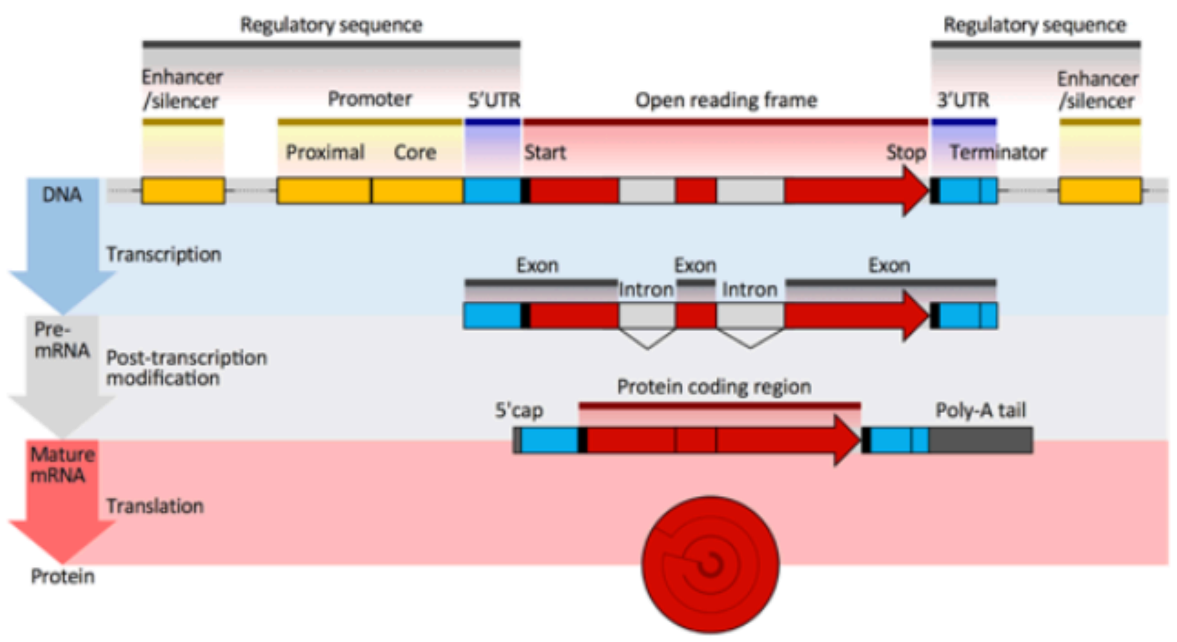
RNA-Seq with Bioconductor in R
by Mary Piper and DataCamp

The Genome
all living organisms contain the instructions for life in their genome
which is present in the nuclei of their cells
the genome is comprised of DNA divided into chromosomes
the building blocks of DNA are called nucleotides
our bases are guanine, adenine, cytosine, and thymine
DNA is double-stranded
forms a helix with a sugar-phosphate backbone
A to T
G to C
the order of these pairs is referred to as the DNA sequence

Genes
with the DNA sequence are regions called genes
genes provide instructions to make proteins
proteins perform some function within the cell
to make proteins, DNA is transcribed into messenger RNA (mRNA)
this is translated by the ribosome into protein
some genes encode RNA that does not get translated into protein called non-coding RNAs (ncRNA)
ncRNAs have a function in and of themselves and include rRNAs, tRNAs, siRNAs, and others
All RNAs transcribed from genes are called transcripts

RNA processing



Wikimedia Commons Gene structure eukaryote 2 annotated.svg by Thomas Shafee, used under Creative Commons Attribution 4.0 International

to be translated into proteins, mRNA must undergo processing
above describes the process
the top strand represents a gene in the DNA
UTR represents the untranslated regions
after post-transcriptional processing, the introns are spliced out and a polyA tal
5′cap are added to yield mature mRNA transcripts
these mature mRNA transcripts can be translted into protein
mRNA transcripts have a polyA tail, which is a sequence of AAAs at the end stating
the end of the transcript
many non-coding RNAs do not have this
we have different cells due to the different genes that are turned on in these cells

Gene expression in disease
a mutation can affect the type and quanity of RNAs and proteins produced
*to explore gene expression changes that occur in disease, it can be useful to
measure the quantity of RNA expressed by all genes
we can do this by using RNA-Seq
then differentail expression analysis of RNA-Seq data can be used to determine
whether there are significant differences in gene expression between conditions

RNA-Seq questions
what genes are differentially expressed between sample groups?
are there any trends in gene expression over time or across conditions?
which groups of genes change similarly over time or across conditions?
what processes or pathways are enriched for my condition of interest?

Need libraries for RNA-Seq workflow
# Load library for DESeq2
library(DESeq2)

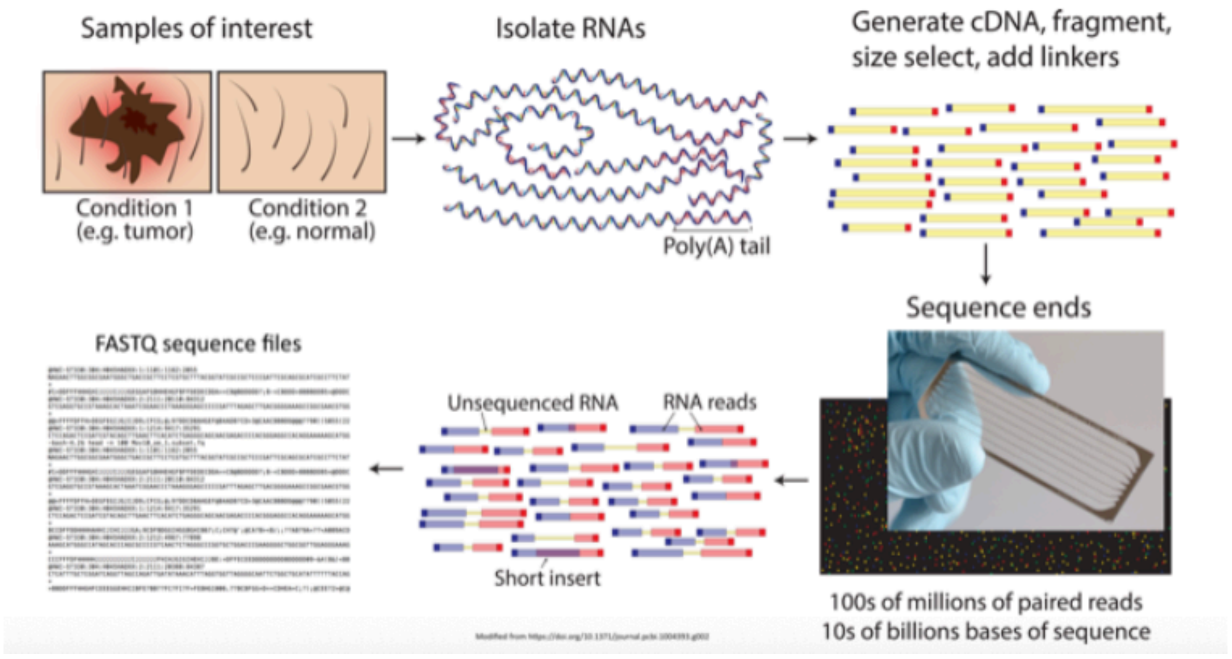
# Load library for RColorBrewer
library(RColorBrewer)

# Load library for pheatmap
library(pheatmap)

# Load library for tidyverse
library(tidyverse)

RNA-Seq Workflow
Experimental Design
'Technical replicates' are of low variation
*'Biological replicates' are key, crucial to the success of analyses
the more the better but at the very least have to have 3
*try to avoid experiments that are performed batch-wise
goal is to perform experiment across all conditions at the same time
if cannot avoid distribute samples from each sample group into each batch
avoid major sources of variation (ie mix sexes)

Workflow

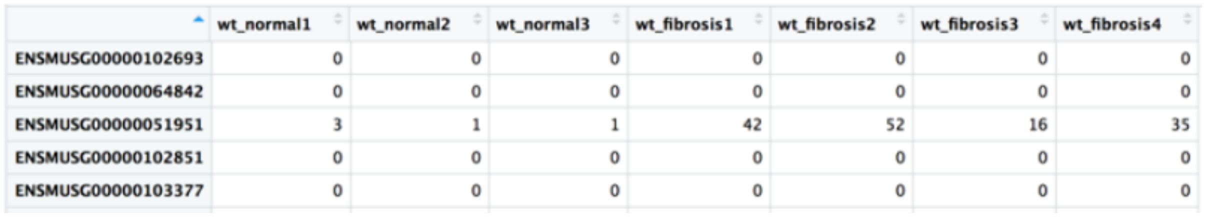

Samples of interest
Condition 1 (e.g. tumor)
Condition 2 (e.g. normal)
Isolate RNAs
Poly(A) tail
Generate cDNA, fragment, size select, add linkers
Sequence ends
100s of millions of paired reads
10s of billions bases of sequence
Unsequenced RNA
RNA reads
Short insert
FASTQ sequence files

RNA isolated and DNA contamination removed

rRNA removed or mature mRNAs are selected by their polyA tails
RNA is turned into cDNA > fragmented, size selected, and adapters are added
done to generate the RNA-Seq libraries to be sequenced
sequencing generates millions of nucleotide sequences called reads
reads correspond to ends of the fragments sequenced
the sequence of each read is output into fastq files
now we assess the quality of the raw data
ensuring nothing went wrong at the sequencing facility
looking for data contamination
next is alignment or mapping of the reads to the genome to determine the location on the genome where the reads originated

More on alignment
mRNA contains only the exons needed to create the proteins
when mRNA is aligned to the genome containing introns, reads will be split across introns
tools for aligning reads to the genome need to align across introns for RNA-seq
ouput of alignment givese the genome coordinates for where the read most likely originated from in the genome

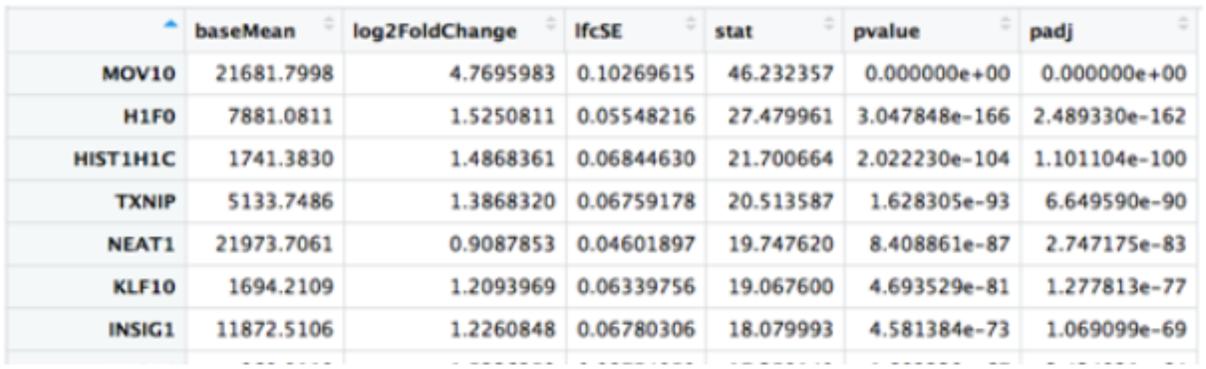
Next the reads aligning to the exons of each gene are quantified to yield a matrix of gene counts
**up to this point we need to use command line tools
from here we can move into using R and Bioconductor
example
we_rawcounts <- read.csv("fibrosis_wt_rawcounts.csv")
this example output > samples as columns and gene IDs as rows



	wt_normal1	wt_normal2	wt_normal3	wt_fibrosis1	wt_fibrosis2	wt_fibrosis3	wt_fibrosis4
ENSMUSG00000102693	0	0	0	0	0	0	0
ENSMUSG00000064842	0	0	0	0	0	0	0
ENSMUSG00000051951	3	1	1	42	52	16	35
ENSMUSG00000102851	0	0	0	0	0	0	0
ENSMUSG00000103377	0	0	0	0	0	0	0

here the count values represent the number of reads or fragments aligning to the exons of each gene

Example of statistical analysis



	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
MOV10	21681.7998	4.7695983	0.10269615	46.232357	0.000000e+00	0.000000e+00
H1F0	7881.0811	1.5250811	0.05548216	27.479961	3.047848e-166	2.489330e-162
HIST1H1C	1741.3830	1.4868361	0.06844630	21.700664	2.022230e-104	1.101104e-100
TXNIP	5133.7486	1.3868320	0.06759178	20.513587	1.628305e-93	6.649590e-90
NEAT1	21973.7061	0.9087853	0.04601897	19.747620	8.408861e-87	2.747175e-83
KLF10	1694.2109	1.2093969	0.06339756	19.067600	4.693529e-81	1.277813e-77
INSIG1	11872.5106	1.2260848	0.06780306	18.079993	4.581384e-73	1.069099e-69

includes log2 fold changes of expression between conditions and the adjusted p-values for each gene
genes that reach a threshold for significance can be subset to define a list of significant differentially expressed genes of potential interest

More on fold change
A fold change is a measure that describes how much a quantity changes between two conditions. In the context of genomics and gene expression studies, a fold change is often used to quantify the difference in expression levels of a gene under two different experimental conditions, such as a treatment group compared to a control group.

The formula for calculating fold change is straightforward:

\[\text{Fold Change} = \frac{\text{Expression in Condition A}}{\text{Expression in Condition B}} \]

Here's how to interpret fold change:

- If the fold change is \(1\), it indicates no change in expression between the two conditions.
- If the fold change is greater than \(1\), it suggests upregulation or an increase in expression in the first condition compared to the second.
- If the fold change is less than \(1\), it suggests downregulation or a decrease in expression in the first condition compared to the second.

For example, a fold change of \(2\) means that the expression level in Condition A is two times higher than in Condition B. Similarly, a fold change of \(0.5\) means that the expression level in Condition A is half that of Condition B.

Fold change is a valuable metric in genomics because it provides a relative measure of the magnitude of differences in gene expression between experimental conditions. Researchers often use fold changes along with statistical tests to identify genes that show significant changes in expression under specific conditions.

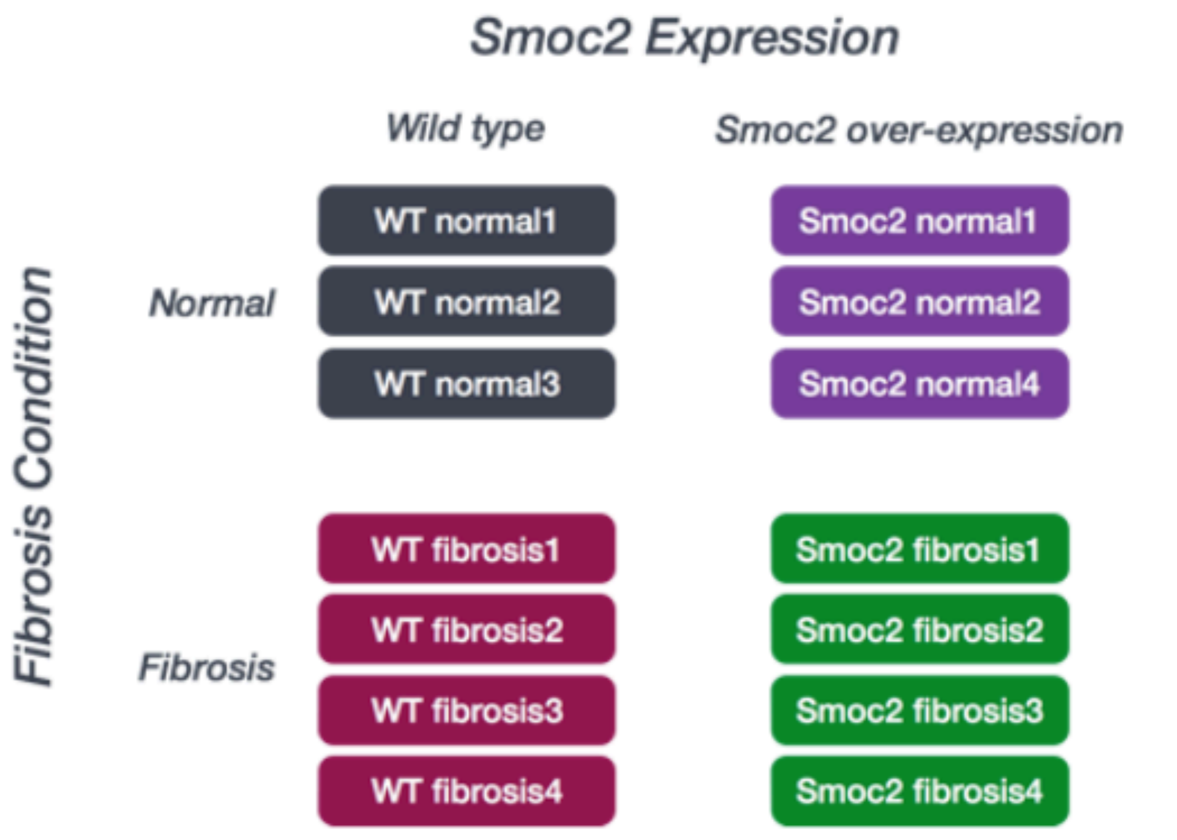
Example

```
# Explore the first six observations of smoc2_rawcounts
head(smoc2_rawcounts)

# Explore the structure of smoc2_rawcounts
str(smoc2_rawcounts)
```

Differential gene expression overview
we use DESeq2 to determine wheter genes are expressed at significantly different levels between two or more sample groups
the question off the bat is always why not just identify the genes with the largest fold changes in expression bewtween sample groups?
the answer is because we need to account for variation
*goal is to determine for each gene whether the differences in expression bewtween groups is significant given the amount of variation within groups (or between the biological replicates)

Example that we are going to work on is the Smoc2 gene
ie Secreted modular calcium-binding protein 2
noted to have increased expression in kidney fibrosis
kidney fibrosis is characterized by an excess of extracellular matrix in the space between tubules and capillaries

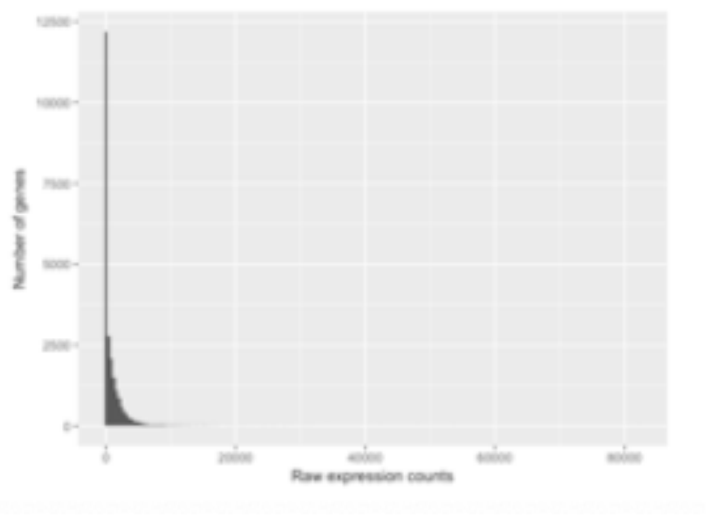
Our dataset



Picking the appropriate statistical model > **this is determined by the count distribution
start by visualizing distribution (here we're using the wild type data)

```
ggplot(raw_counts) +
	geom_histogram(aes(x = wt_normal1), stat = 'bin', bins = 200) +
	xlab("Raw expression counts") +
	ylab("Number of genes")
```

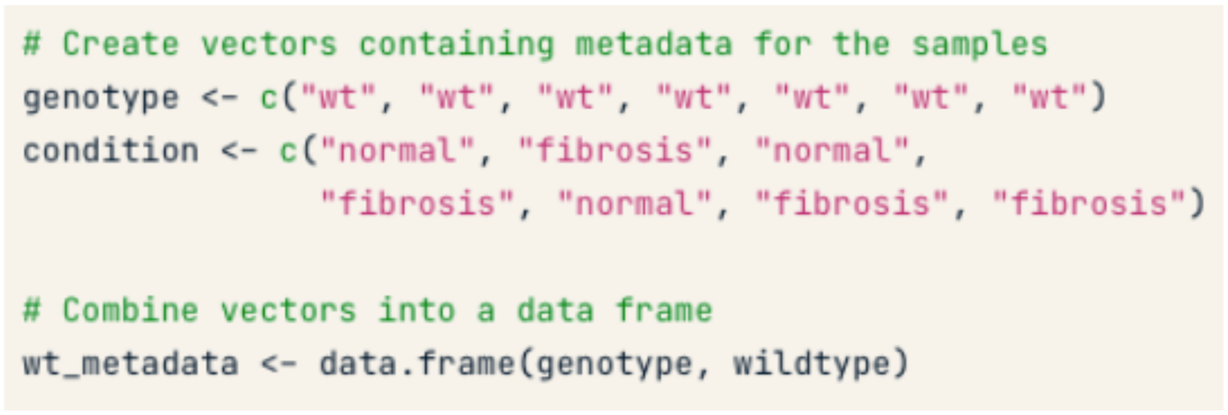
output>



noted long right tail (this is due to there being no limit for maximu expression in RNA-Seq data)
looks like a Poisson distribution
however since there is expression variation between biological replicates Poisson would not be an appropriate model
*here the best model to use is the negative binomial model

Prep for DESeq2 object
dds <- DESeqDataSetFromMatrix(countData = rawcounts,
                                        colData = metadata,
                                        design = ~condition)
*the design formula given should contain major expected sources of variation to control for and the condition of interest as the last term in the formula

The metadata



```
# Create vectors containing metadata for the samples
genotype <- c("wt", "wt", "wt", "wt", "wt", "wt", "wt")
condition <- c("normal", "fibrosis", "normal",
               "fibrosis", "normal", "fibrosis", "fibrosis")

# Combine vectors into a data frame
wt_metadata <- data.frame(genotype, wildtype)
```

at the very least we need to know which of our samples correspond to each condition
sample names are added as the row names

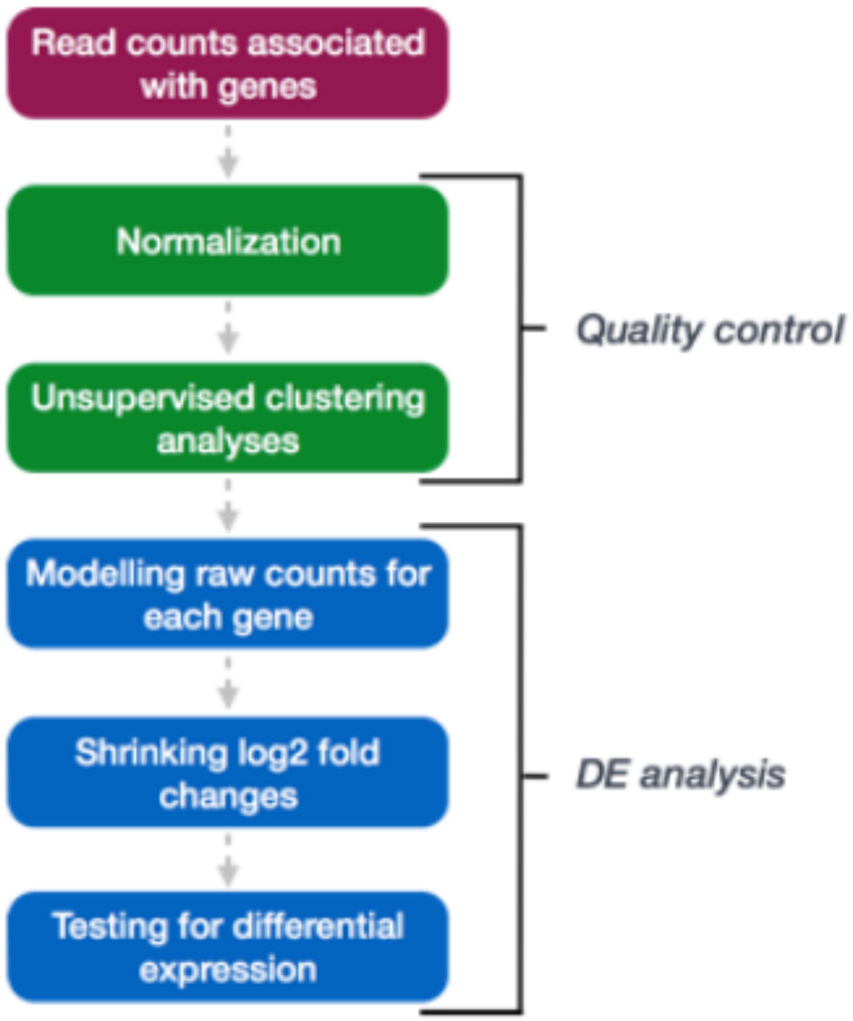
Example

```
# Create genotype vector
genotype <- c("smoc2_oe", "smoc2_oe", "smoc2_oe", "smoc2_oe", "smoc2_oe",
"smoc2_oe", "smoc2_oe")

# Create condition vector
condition <- c("fibrosis", "fibrosis", "fibrosis", "fibrosis", "normal", "normal",
"normal")

# Create data frame
smoc2_metadata <- data.frame(genotype, condition)

# Assign the row names of the data frame
rownames(smoc2_metadata) <- c("smoc2_fibrosis1", "smoc2_fibrosis2",
"smoc2_fibrosis3", "smoc2_fibrosis4", "smoc2_normal1", "smoc2_normal3",
"smoc2_normal4")
```

Using DESeq2
a great function to get detailed info is;

```
vignette(DESeq2)
```

Workflow


Read counts associated with genes
Normalization
Unsupervised clustering analyses
Quality control
Modelling raw counts for each gene
Shrinking log2 fold changes
Testing for differential expression
DE analysis

first we normalize counts to account for differences in library depth
then we use PCA and hierarchical clustering using heatmaps to identify potential sample outliers and major variations in the data
then the DE analysis

Bringing data in
metadata and counts datasets need to be in the same order
row names of our metadata need to be in the same order as the column names of our counts data
can explore this data easily with these functions:
rownames(wt_metadata)
colnames(wt_rawcounts)
easy way to check if our rows and columns are in proper order >
all(rownames(wt_meatdata) == colnames(wt_rawcounts))
output witll be a logical response

How to match them?

```
match(vector1, vector2)
#vector1 represents the set with the desired order
#vector2 represents the set of values to reorder
```

ouput > the indices for how to rearrang vector2 to be in the same order as vector1
now we can reorder
example

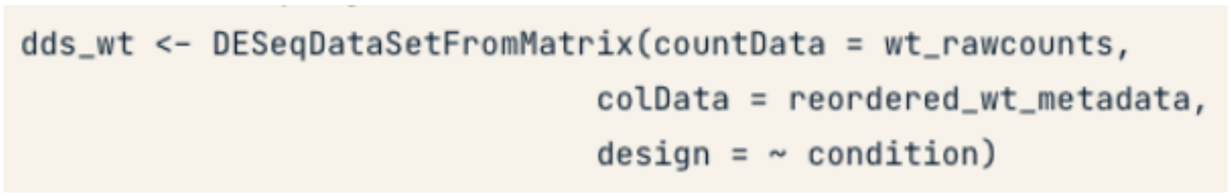
```
idx <- match(colnames(wt_rawcoutns), rownames(wt_metadata))
reordered_wt_metadata <- wt_metadata[idx, ]
#confirm in proper order
View(reordered_wt_metadata)
```

and

```
all(rownames(wt_meatdata) == colnames(wt_rawcounts))
```

output shall now be TRUE

Create the DESeq2 object



```
dds_wt <- DESeqDataSetFromMatrix(countData = wt_rawcounts,
                                 colData = reordered_wt_metadata,
                                 design = ~ condition)
```

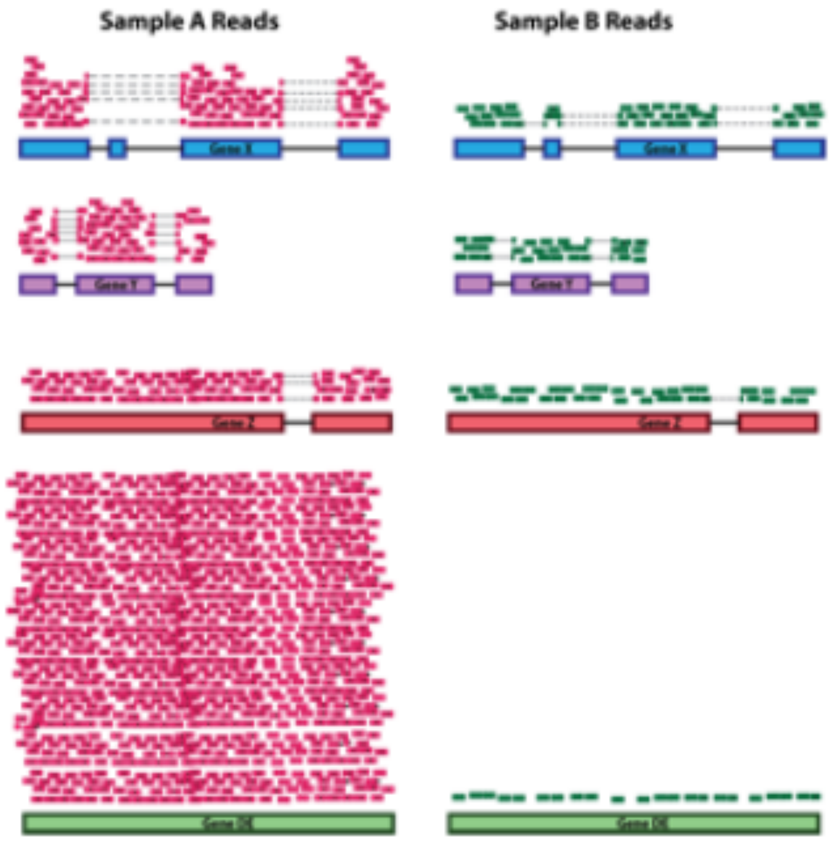
this object has class Ranged Summarized Experiment
this is a list-like object with slots available for the data it will generate throughout the analysis

Example

```
# Use the match() function to reorder the columns of the raw counts
reorder_idx <- match(rownames(smoc2_metadata), colnames(smoc2_rawcounts))

# Reorder the columns of the count data
reordered_smoc2_rawcounts <- smoc2_rawcounts[ , reorder_idx]

# Create a DESeq2 object
dds_smoc2 <- DESeqDataSetFromMatrix(countData = 
reordered_smoc2_rawcounts,
                          colData =  smoc2_metadata,
                          design = ~ condition)
```

Count normalization
raw counts represent the number of reads aligning to each gene and should be proportional to the expression of the RNA in the sample
factors other than the expression of the RNA can influence the number of reads aligning to each gene

normalization methods can remove the influence of these factors
factors considered are library depth, gene length, and RNA composition
*DE analysis compares expression levels of the same genes between conditions, we do not need to normalize for gene length
*however to compare the expression levels of different genes, you would need to account for lengths of the genes
*also be aware that a few highly differentially expressed genes can skew many normalization methods
example gene DE



to get around this DESeq2 uses a ‘median of ratios’ method of normalization
this method adjusts the raw counts for library size and is resistant to large numbers of differentially expressed genes

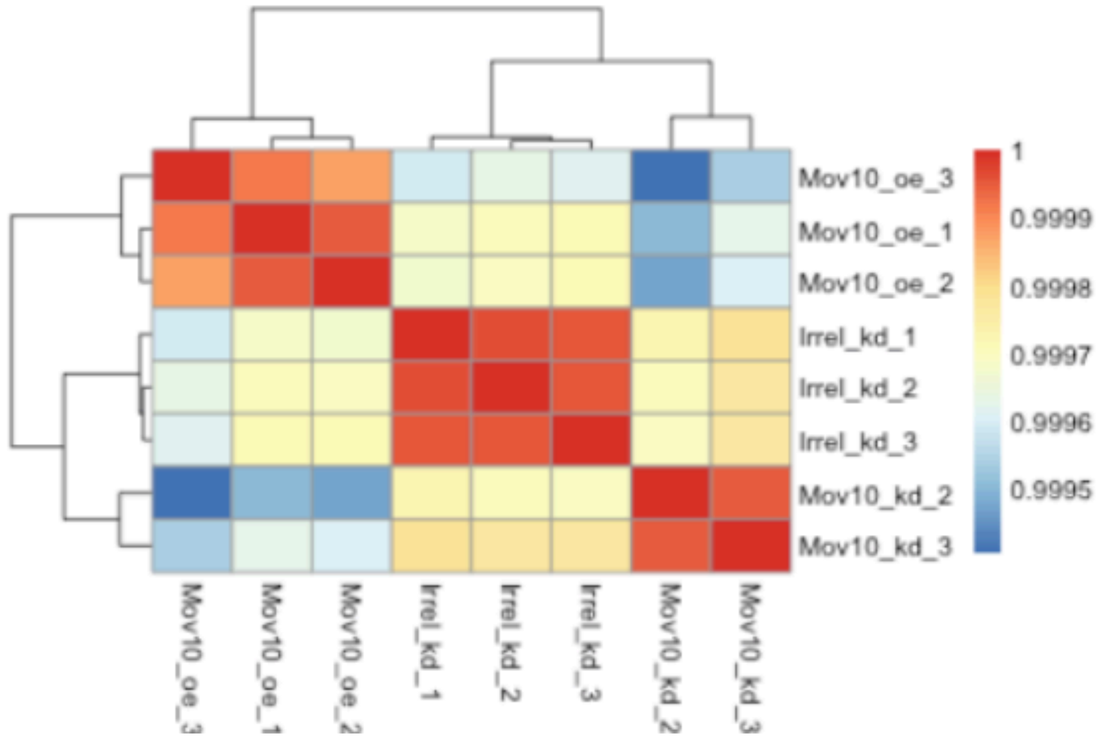
To calculate
dds_wt <- estimateSizeFactors(dds_wt)
*this output gets assigned to a slot in the DESeq2 object dds_wt
the raw counts for each sample are divided by the associated sample-spcecific size factor for normalization
sizeFactors(dds_wt) #lets us view the size factors used for normalization

Now we can extract

we do this with the counts() function
normalized_wt_counts <- counts(dds_wt, normalized=TRUE)
ensure to put 'normalized' to true or else you will get the raw counts

Hierarchical heatmap
a solid tool to explore how similar the samples are to each other
first we need to log transform the normalized counts to improve the visualization of the clustering
DESeq2 uses a variance stabilizing transformation
this moderates the variance across the mean
'blind' argument specifies that the transformation should be blind to the sample information given in the design formula
vsd_wt <- vst(dds_wt, blind=TRUE)

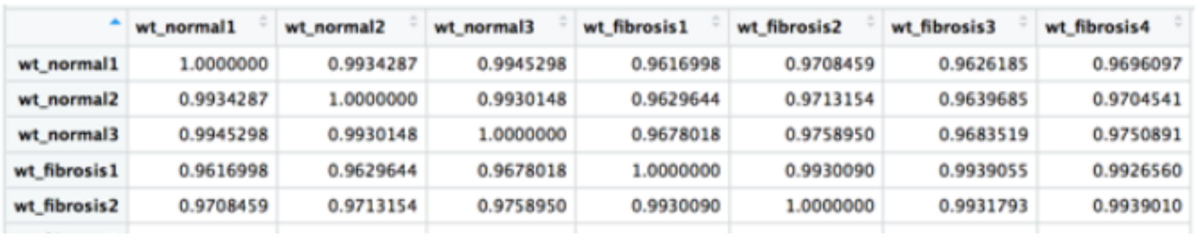
Heatmap example



assesses the similarity in gene expression between the different samples in a dataset
used to explore how similar replicates are to each other and whether the samples belonging to different sample groups cluster separately
the heatmap is created by using the gene expression correlation values for all pairwise combinations of samples in the dataset
with 1 being a perfect correlation
*the hierarchical tree on the fringes shows which samples are more similar to each other

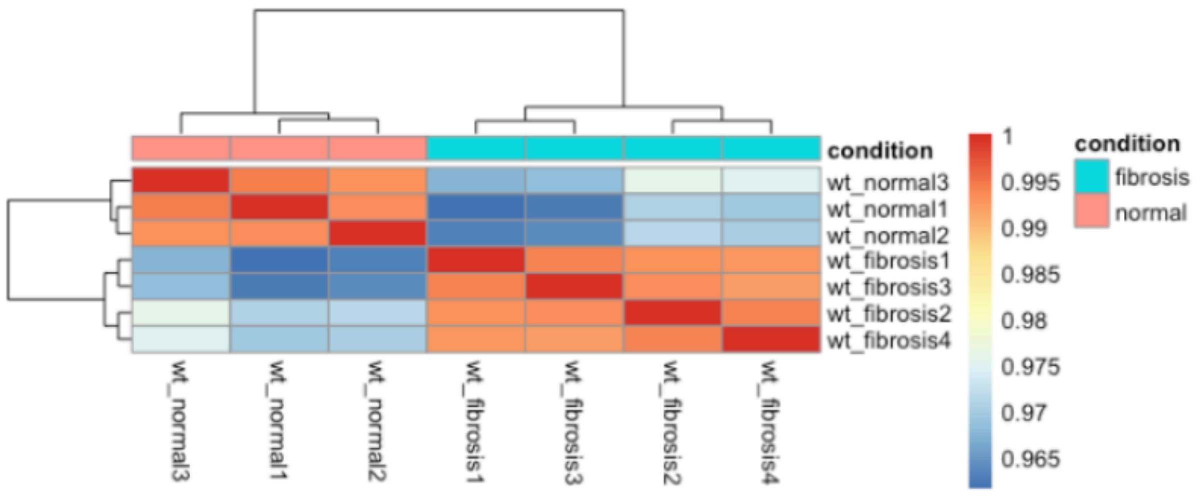
*we expect the biological replicates to cluster together and sample conditions to cluster apart
majority of genes should not be differently expressed, so samples should generally have high correlations with each other
*samples with correlation values below 0.8 require further investigation for contamination or outliers

to create a hierarchical heatmap
#extract the vst-normalized counts as a matrix from the vsd object
vsd_mat_wt <- assay(vsd_wt)
#then compute the pairwise correlation values between each pair of samples using the cor() function
vsd_cor_wt <- cor(vsd_mat_wt)
View(vsd_cor_wt)
output>



	wt_normal1	wt_normal2	wt_normal3	wt_fibrosis1	wt_fibrosis2	wt_fibrosis3	wt_fibrosis4
wt_normal1	1.0000000	0.9934287	0.9945298	0.9616998	0.9708459	0.9626185	0.9696097
wt_normal2	0.9934287	1.0000000	0.9930148	0.9629644	0.9713154	0.9639685	0.9704541
wt_normal3	0.9945298	0.9930148	1.0000000	0.9678018	0.9758950	0.9683519	0.9750891
wt_fibrosis1	0.9616998	0.9629644	0.9678018	1.0000000	0.9930090	0.9939055	0.9926560
wt_fibrosis2	0.9708459	0.9713154	0.9758950	0.9930090	1.0000000	0.9931793	0.9939010

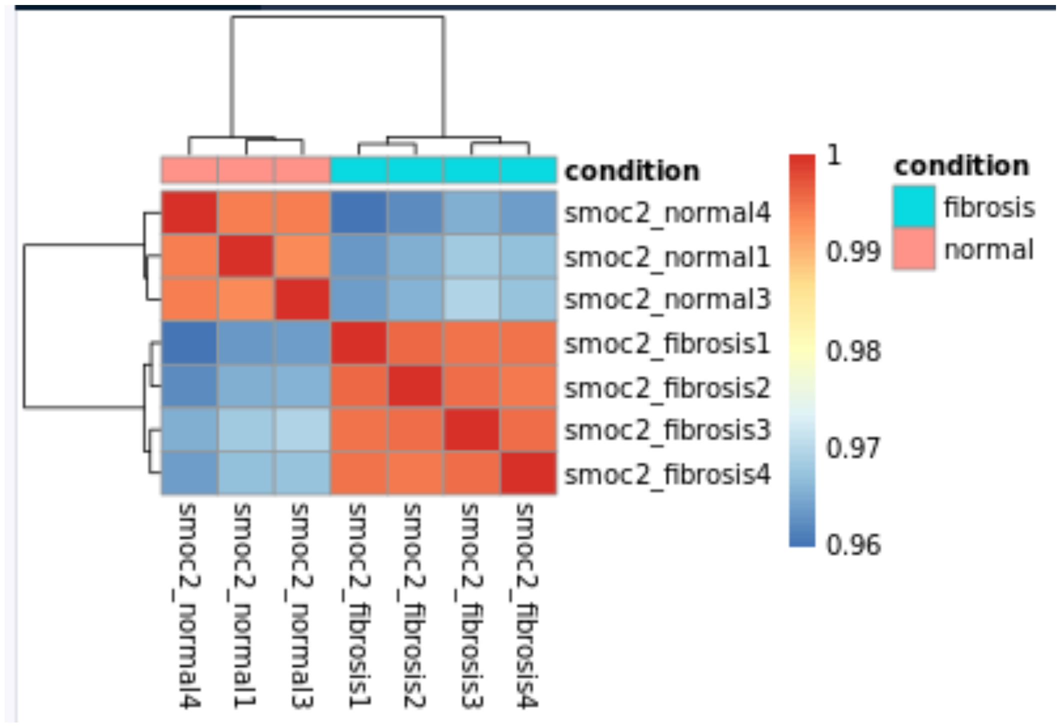
Next
library(pheatmap)
'annotation' argument selects which factors in the metadata to include as annotation bars
use the select() function from the dplyr package to select the condition column in the wildtype metadata
pheatmap(vsd_cor_wt, annotation = select(wt_metadata, condition))
output for our example>



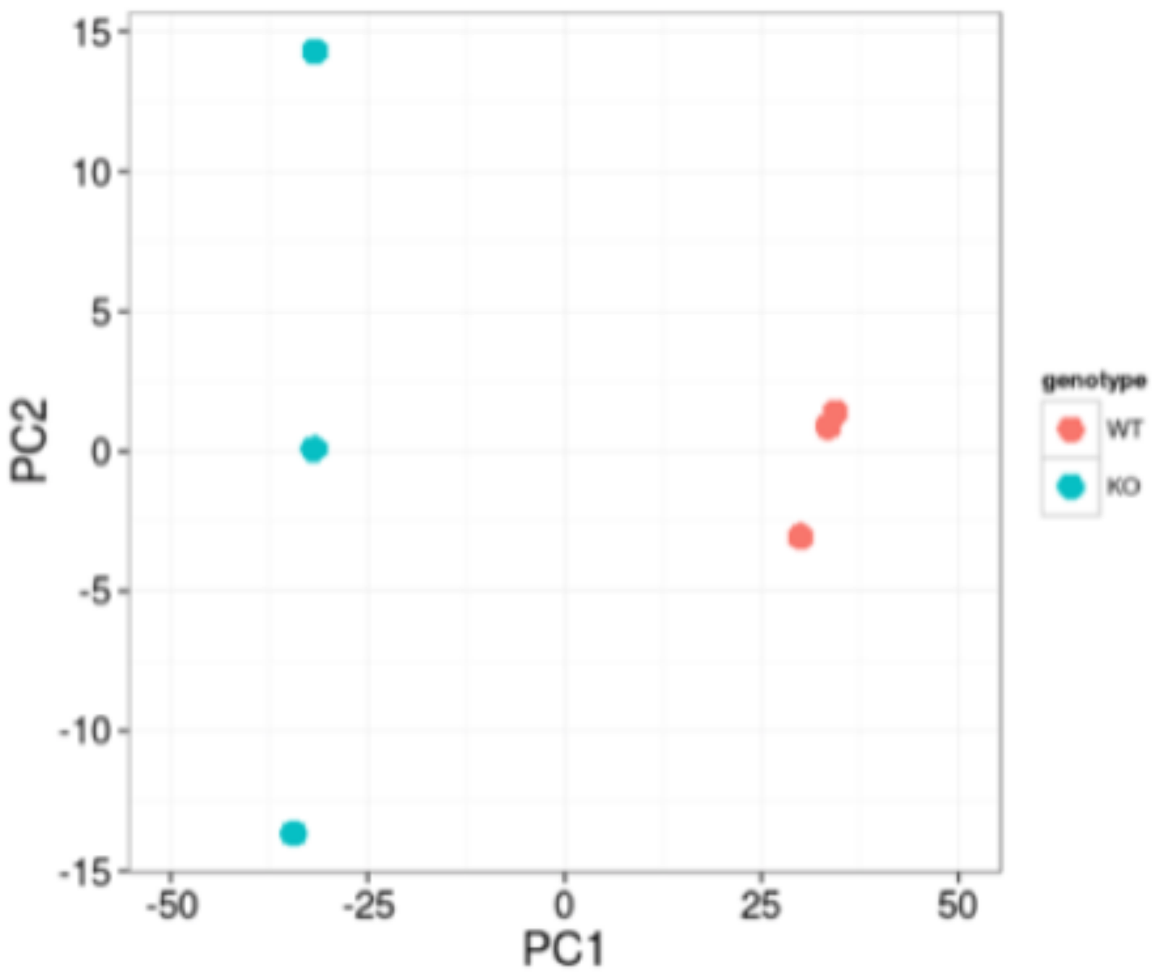
Example

```
# Transform the normalized counts
vsd_smoc2 <- vst(dds_smoc2, blind = TRUE)

# Extract the matrix of transformed counts
vsd_mat_smoc2 <- assay(vsd_smoc2)

# Compute the correlation values between samples
vsd_cor_smoc2 <- cor(vsd_mat_smoc2)

# Plot the heatmap
pheatmap(vsd_cor_smoc2, annotation = select(smoc2_metadata, condition))
```

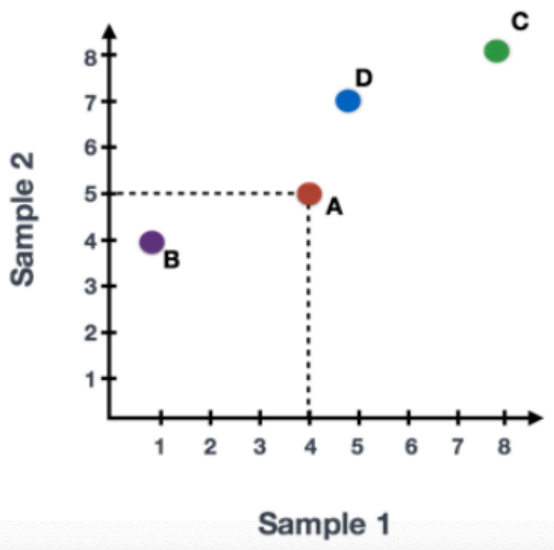
output>

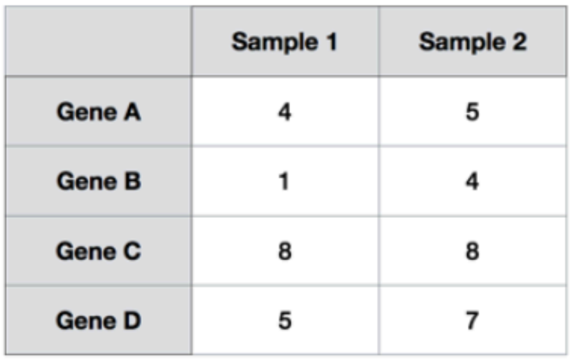


PCA
a technique used to emphasize the variation present in a dataset
find the principal components of a dataset
with PC1 (the first principal component) representing the greatest amount of variance in the data



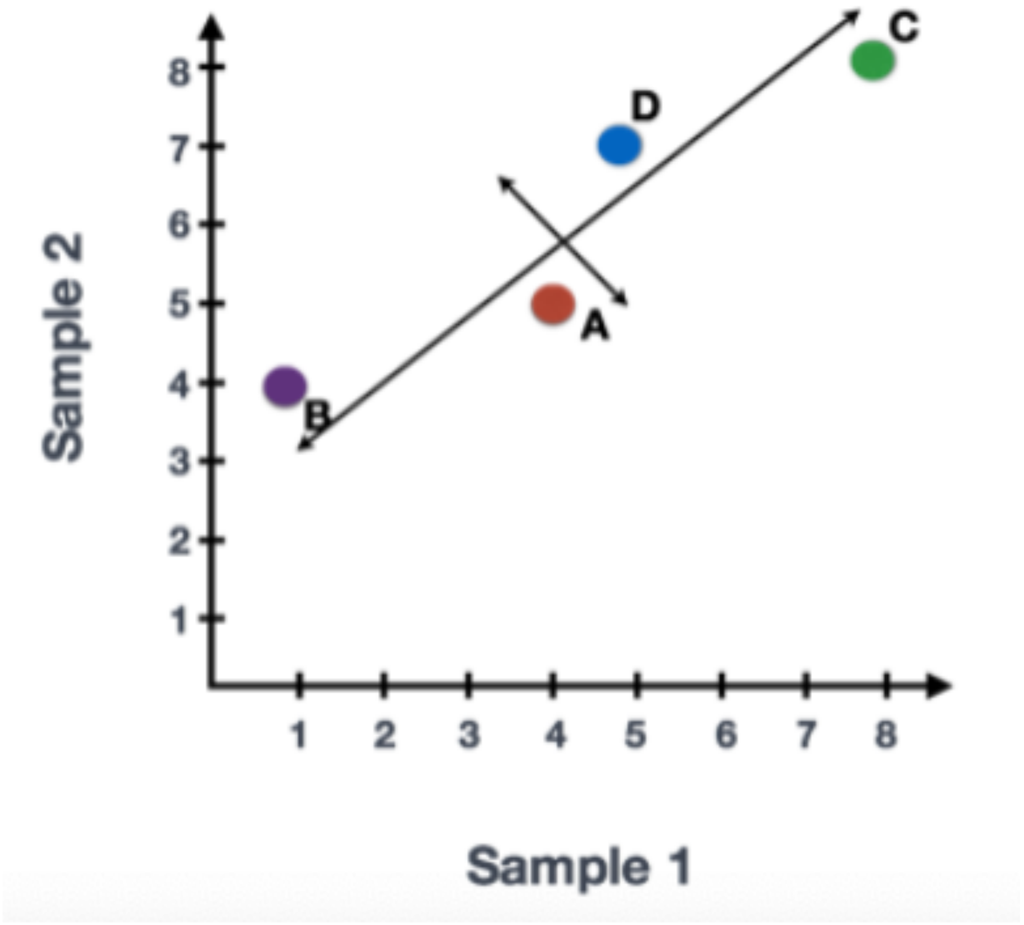
Simple example where we have two samples and we plot the occurence of the particular gene using the x-axis for 1 sample and the y-axis for the other sample



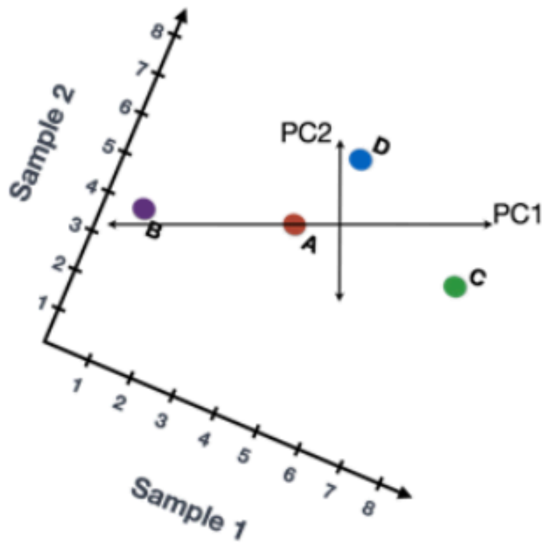


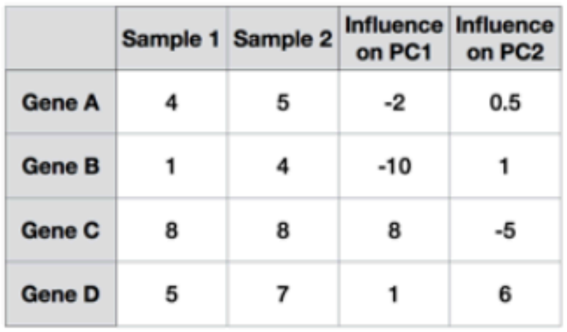
	Sample 1	Sample 2
Gene A	4	5
Gene B	1	4
Gene C	8	8
Gene D	5	7

Next we draw a line where there is the most variation (PC1)



the second most variation (PC2) must be perpendicular to PC1
this is to best describe the variance within the dataset
this is a very simple example, most datasets will be multi-dimensional
*the number of principal components is equal to the number of samples (n) in the dataset
the most variant genes for a principal component have the most influence on that principal component’s direction
quantitative scores to genes based on how much they influence the different PCs





	Sample 1	Sample 2	Influence on PC1	Influence on PC2
Gene A	4	5	-2	0.5
Gene B	1	4	-10	1
Gene C	8	8	8	-5
Gene D	5	7	1	6

per sample PC value is computed by taking the product of the influence and the normalized read count for each gene and summing across all genes

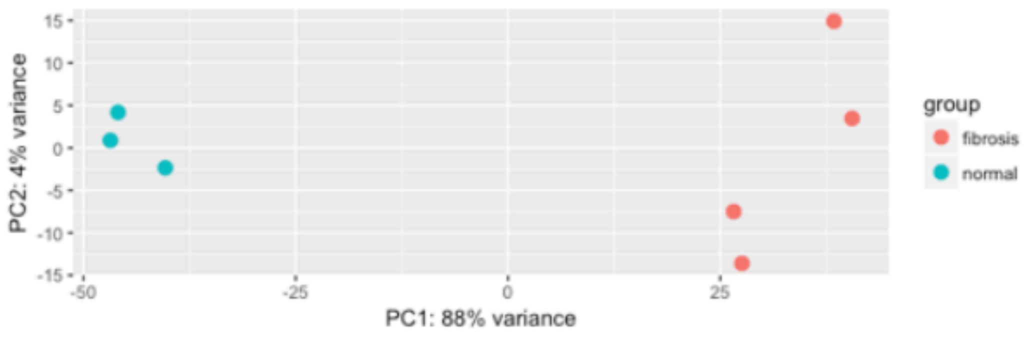
```
Sample1 PC1 score = (4 * -2) + (1 * -10) + (8 * 8) + (5 * 1) = 51
Sample1 PC2 score = (4 * 0.5) + (1 * 1) + (8 * -5) + (5 * 6) = -7

Sample2 PC1 score = (5 * -2) + (4 * -10) + (8 * 8) + (7 * 1) = 21
Sample2 PC2 score = (5 * 0.5) + (4 * 1) + (8 * -5) + (7 * 6) = 8.5
```

we then plot these per sample PC values
samples that cluster together have more similar gene expression profiles than samples that cluster apart

PCA can help identify sample outliers and major sources of variation

Using DESeq2
#plotPCA() takes as an input the transformed vsd object
#’intgroup’ argument specifies what factor in the metadata to use to color the plot

```r
# Plot PCA
plotPCA(vsd_wt, intgroup="condition")
```



what this shows
on PC1 we can see that the sample groups fibrosis and normal separate nicely
this means that our condition corresponds to PC1 representing 88% of the variance in our data
4% of the variance is explained in PC2
basically saying the majority of the variation in gene expressin in the dataset can likely be explained by the differences between sample groups
if they didn’t separate that could mean that the effect of the condition could be small
or other larger sources of variation may be present
can color other factors (ie age, sex,…) to identify these
also function pcomp() allows a more thorough analysis of PCs other than PC1 and PC2

DE analysis
reminder first steps (from above)
explore quality of samples
remove any outlier samples
assess the major sources of variation present

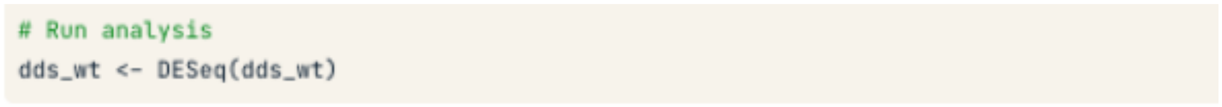
DE analysis stepes
1. fitting the raw counts for each gene to the DESeq2 negative binomial model and testing for differential expression
2. shrinking the log2 fold changes
3. extracting and visualizing the results

Design formula
tells DESeq2 the known major sources of variation to control for (or regress out) and the condition of interest to use for differential expression testing

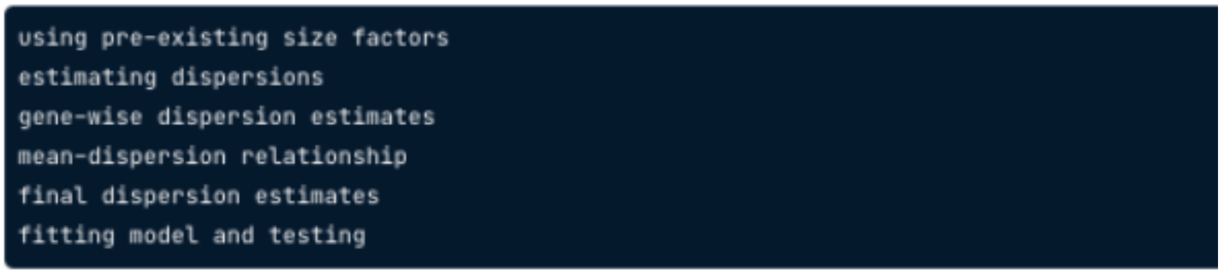
*condition of interest comes last in our design formula
example from above
dds_wt <- DESeqDataSetFromMatrix(countData = wt_rawcounts,
                                                    colData = reordered_wt_metadata,
                                                    design = ~ strain + sex + treatment)
#here treatment is our condition of interest
#the other factors are placed because in our prior assessment they proved to be major sources of variation
#these other factors can be placed in any order but the condition of interest needs to be last
tilde tells DESeq2 to model the counts using the following formula
factor names in the design formula need to exactly match the column names in the metadata

We can do more complex designs with DESeq2
example say we wanted to know the effect of sex on the effect of treatment we could use an interaction term
~ strain + sex + treatment + sex:treatment

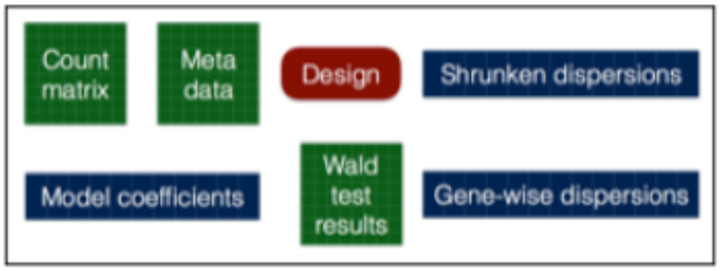
Now we can perform model fitting
example



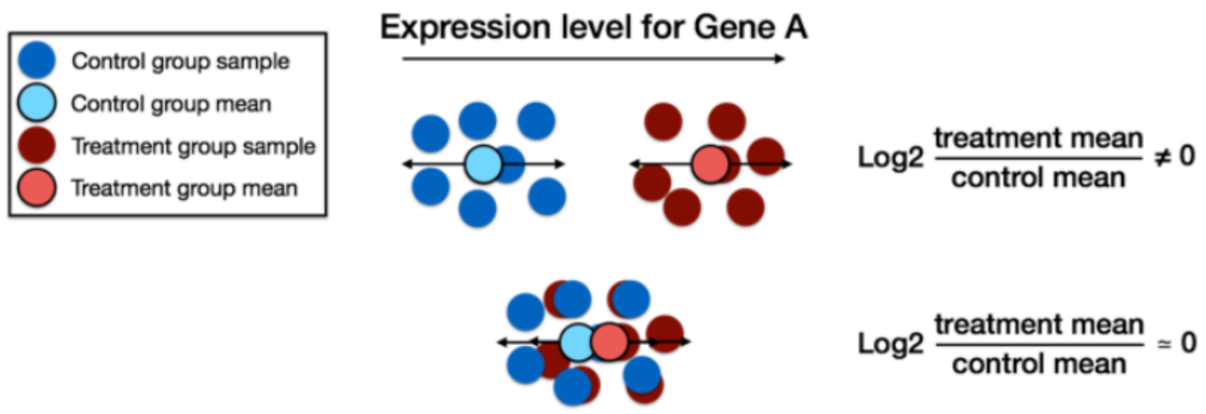
```
# Run analysis
dds_wt <- DESeq(dds_wt)
```



```
using pre-existing size factors
estimating dispersions
gene-wise dispersion estimates
mean-dispersion relationship
final dispersion estimates
fitting model and testing
```



Assessing how well the model fits


Control group sample
Control group mean
Treatment group sample
Treatment group mean
Expression level for Gene A
Log2 treatment mean / control mean ≠ 0
Log2 treatment mean / control mean ≈ 0

the goal of the differential expression analysis is to determine wheter a gene's mean expression between sample groups is different given the variation within groups
determined by testing the probability of the log2 fold changes between groups being significantly different from zero
to model the counts requires information about the mean and variation in the data
to explore this variation we observe the variance in gene expression relative to the mean
variance (square of std) represents how far away the expression of the individual samples are from the means

For RNA-Seq data
variance is generally expected to increase with the gene's mean expression
the apply() function allows us to calculate the means and variances for every gene of the normal samples

```
# Syntax for apply()
apply(data, rows/columns, function_to_apply)
```

```
# Calculating mean for each gene (each row)
mean_counts <- apply(wt_rawcounts[, 1:3], 1, mean)
```

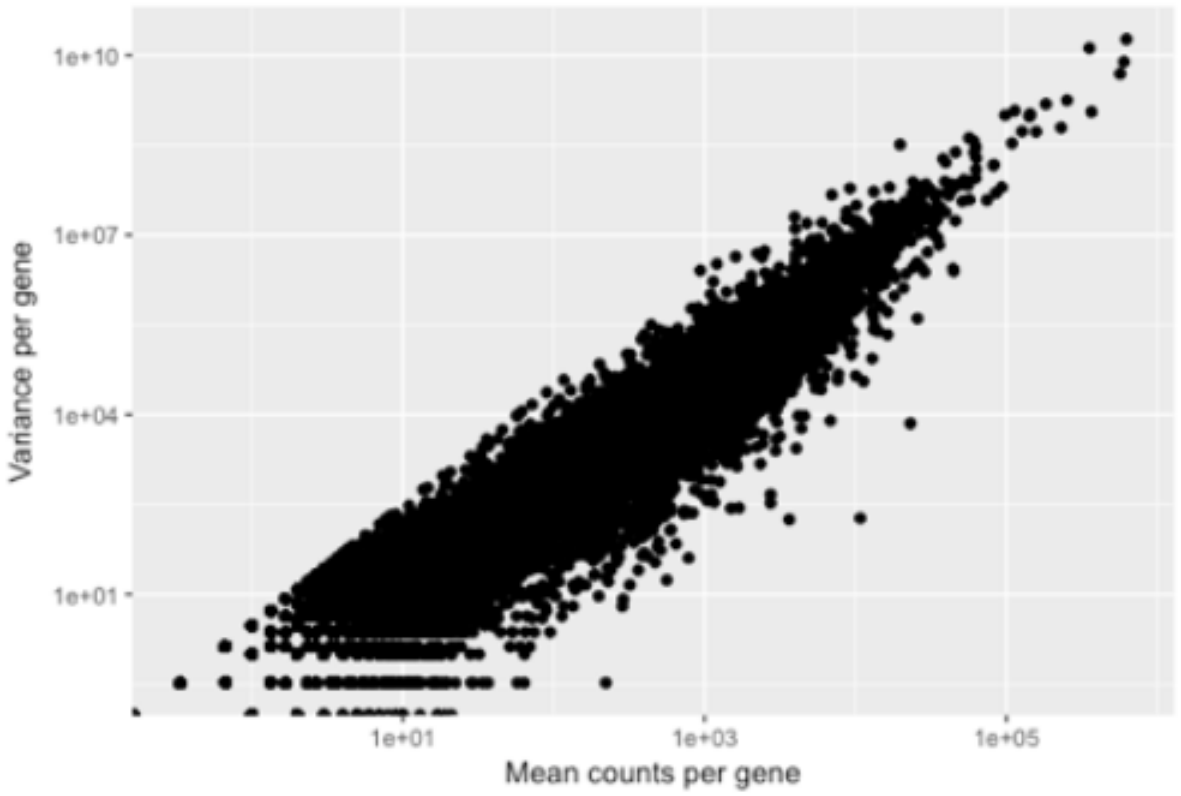
```
# Calculating variance for each gene (each row)
variance_counts <- apply(wt_rawcounts[, 1:3], 1, var)
```

then create a dataframe
then plot

```r
# Creating data frame with mean and variance for every gene
df <- data.frame(mean_counts, variance_counts)
```

```r
ggplot(df) +
        geom_point(aes(x=mean_counts, y=variance_counts)) +
        scale_y_log10() +
        scale_x_log10() +
        xlab("Mean counts per gene") +
        ylab("Variance per gene")
```

a log10 scale helps in visualizing this range more effectively
each black dot represents a gene



the variance in gene expression increase with the mean
this is expected in RNA-Seq data
the range in values for variance is greater for lower mean counts compared to higher mean counts
also expected
‘dispersion’ is a measure of the variance for a given mean

DESeq2 uses dispersion to assess the variability in expression when modeling the counts

Var: variance

μ: mean

α: dispersion

Dispersion formula: $Var = \mu + \alpha * \mu^2$

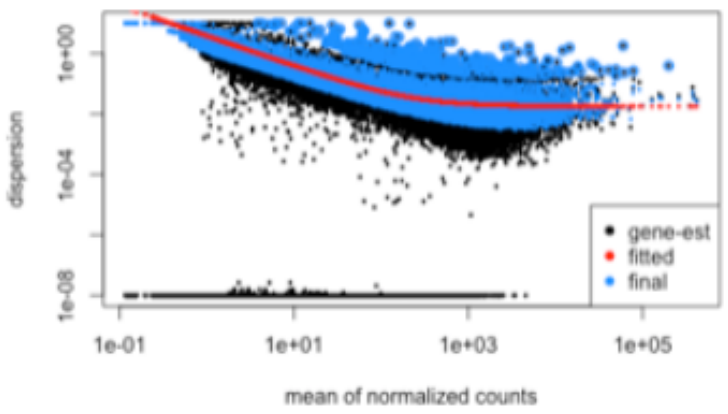
Relationship between mean, variance and dispersion:

$$\uparrow variance \Rightarrow \uparrow dispersion$$

$$\uparrow mean \Rightarrow \downarrow dispersion$$

for any two genes with the same mean expression, the only difference in dispersion will be based on differences in variance
*to check fit of our data to our DESeq2 model, it can be useful to look at the dispersion estimates

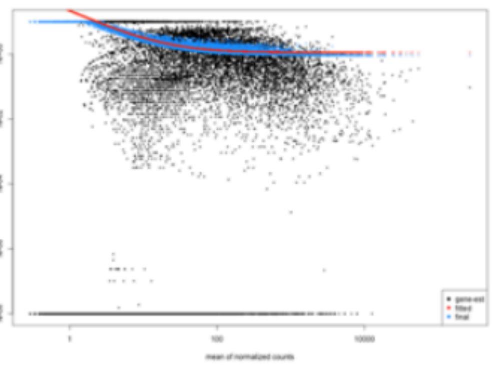
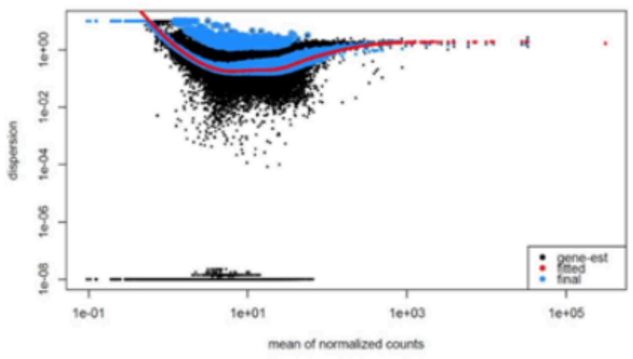
```
# Plot dispersion estimates
plotDispEsts(dds_wt)
```



each black dot is a gene with associated mean and dispersion values
we expect dispersion values to decrease with increasing mean, which is what we

see
few replicates can mean gene-wise estimates of dispersion are often inaccurate
DESeq2 uses infromation across all genes to determine the most likely estimates of dispersion #red line
*genes with inaccurately small estimates of variation could yield many false positives or genes that are identified as DE (when they are not)
the black dots are the originals
the blue dots are the originals shrunken towards the curve
this yields more accurate estimates of dispersion
the blue dot estimates of dispersion are used to model the counts for determining the differentially-expressed genes
*extremely high dispersion values (blue circles with black dots inside) are not shrunke
this is due to the likelihood the the gene may have higher variability than others (biological or technical reasons)
*and reducing this variation could result in false positives
larger number of replicates can estimate the mean and variation more accurately so yield less shrinkage

Worrisome dispersion plots

the first does not follow the curve
the second dispersion does not decrease with increasing mean

DESeq2 Negative Binomial Model

$$\overset{\text{raw count for gene i, sample j}}{K_{ij}} \sim \mathrm{NB}(\overbrace{s_{ij}q_{ij}}^{\text{The mean is taken as “normalized counts” scaled by a normalization factor}}, \overset{\text{one dispersion per gene}}{\alpha_i})$$

negative binomial model accounts fro the additional variation in the data added by the small number of biological replicates
size factors indicated by Sij
normalized counts indicated by Qij
shrunken dispersions denoted by alphai
all used as input to the negative binomial model to fit the raw count data

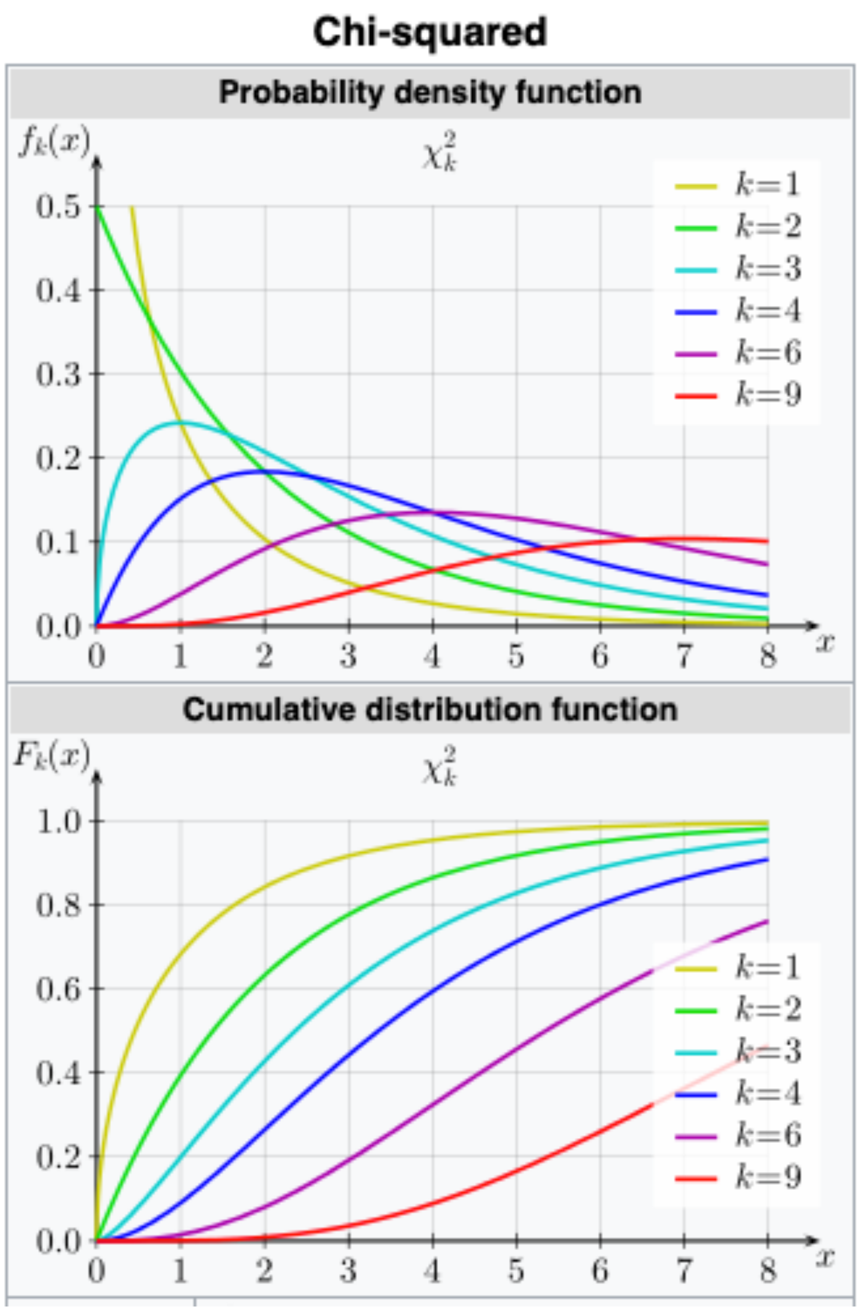
$$\log2 \overset{\text{normalized counts for gene i, sample j}}{q_{ij}} = \Sigma_r x_{jr} \overset{\text{log2 fold change between conditions}}{\beta_{ir}}$$

for each gene the model uses the log2 normalized counts
to determine the log2 foldchange estimates (betair) for the samples of the condition of interest (Xjr)
in addition the associated standard error is also the ouput

DESeq2 contrasts
by default DESeq2 performs the Wald test for pairwise comparisons to test for differences in expression between two sample groups for the condition of interest
the Wald test is a statistical test used to assess the significance of individual coefficients (parameters) in a statistical model
particularly in the context of regression analysis

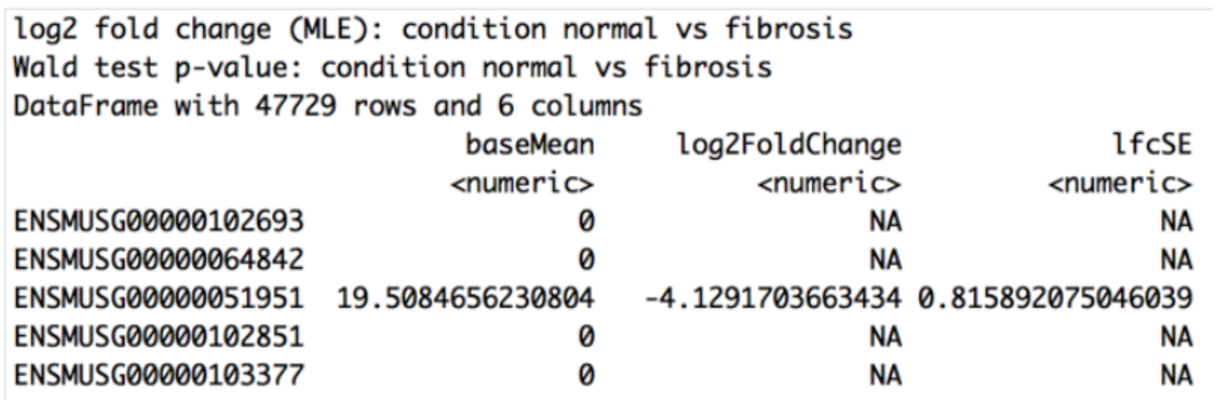
$$Wald\ statistic = \left(\frac{\text{Estimated Coefficient} - \text{Hypothesized Value}}{\text{Standard Error of the Coefficient}}\right)^2$$

with large sample sizes Wald statistic follows a chi-squared distribution
chi-squared distribution is a probability distribution used in hypothesis testing and confidence interval construction
looks like:



chi-squared distribution is positively skewed and takes only non-negative values
characterized by degrees of freedom (denoted as k or v)
this determines the shape of the distribution
if assumed standard normal random variables Z (mean 0 and variance 1), the random variable X follows a chi-squared distribution
as the dofs increase the chi-squared distribution approaches a normal distribution
CDF is used to calculate the prob that a chi-squared random variable is less than or equal to a specified value
Within DESeq2 to get the results of this testing:
results(wt_dds, alpha = 0.05)

alpha is chosen based on how stringent you want to be with your analysis
lower alpha indicate less probability of identifying a gene as DE when it is actually not
output example with condtion meaning normal vs fibrosis>



```
log2 fold change (MLE): condition normal vs fibrosis
Wald test p-value: condition normal vs fibrosis
DataFrame with 47729 rows and 6 columns
                           baseMean     log2FoldChange              lfcSE
                          <numeric>          <numeric>          <numeric>
ENSMUSG00000102693                0                 NA                 NA
ENSMUSG00000064842                0                 NA                 NA
ENSMUSG00000051951  19.5084656230804   -4.1291703663434 0.815892075046039
ENSMUSG00000102851                0                 NA                 NA
ENSMUSG00000103377                0                 NA                 NA
```

this represents the normal group relative to the fibrosis group

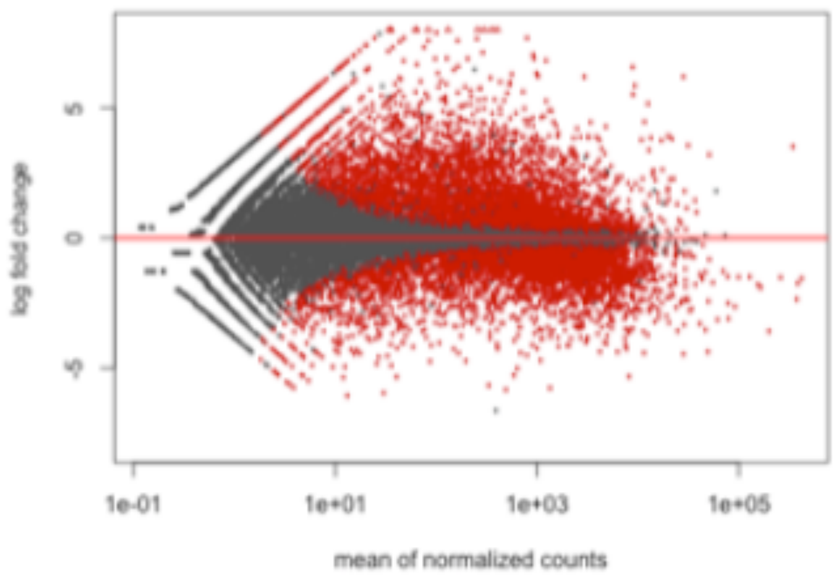
We can perform any pairwise comparison between the sample groups by supplying our own contrast

```
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)
```

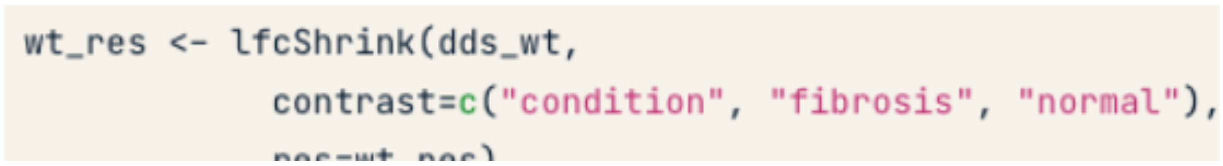
here we define 'normal' as the base level and 'fibrosis' as the level to compare

Exploring these results a bit further
MA plot shows the mean of the normalized counts versus the log2 fold changes for all genes tested
plotMA(wt_res, ylim=c(-8,8))
output>



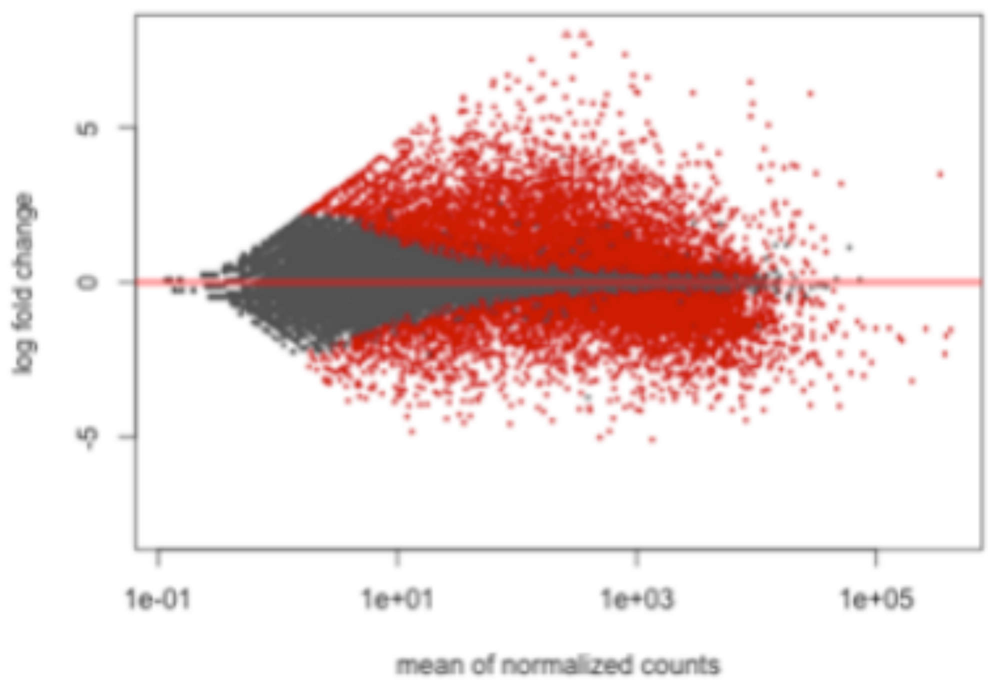
genes that are significantly DE are colored red
large log2 foldchanges for genes with lower mean count values
*these fold changes are unlikely to be as accurate for genes that have little information associated with them
such as genes with low numbers of counts or high dispersion values
refresher on fold changes:
represent the ratio of one value to another
tells you how many times one quantity is larger or smaller than another
Fold Change = B/A
basically says “how many times is B different than A?”
if 1, then quantities are the same
if 2, then B is twice as large as A
if <1, then B is smaller than A

LFC shrinkage
to improve the estimated fold changes
for genes with low amounts of information available, shrinkage use info from all genes to generate more likley (lower) log2 fold changes



```
wt_res <- lfcShrink(dds_wt,
                    contrast=c("condition", "fibrosis", "normal"),
                    res=wt_res)
```

now using the plot MA

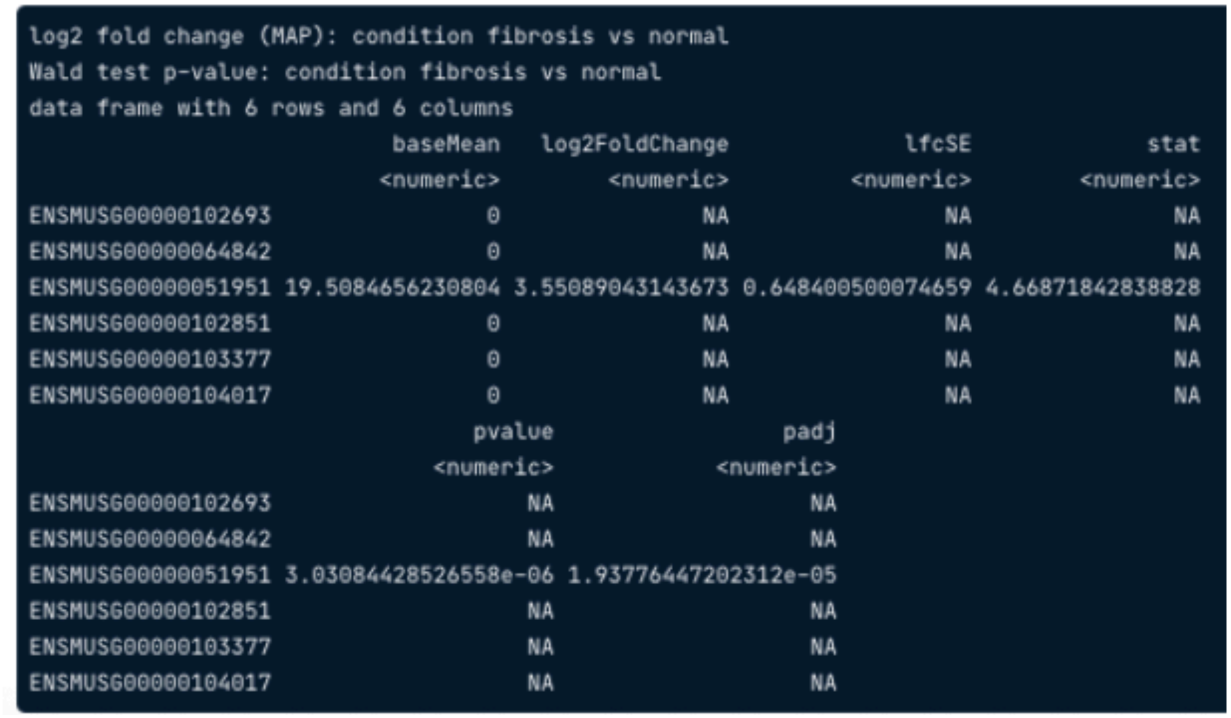


we can see more restricted log2 fold chonge values particularly for lowly expressed genes
these fold changes should be more accurate
*this will not affec the number of differentially expressed genes returned
now we can extract the significant DE genes and perform further visualization of results

Example

```
# Shrink the log2 fold change estimates to be more accurate
smoc2_res <- lfcShrink(dds_smoc2,
                contrast =  c("condition", "fibrosis", "normal"),
                res = smoc2_res)
```

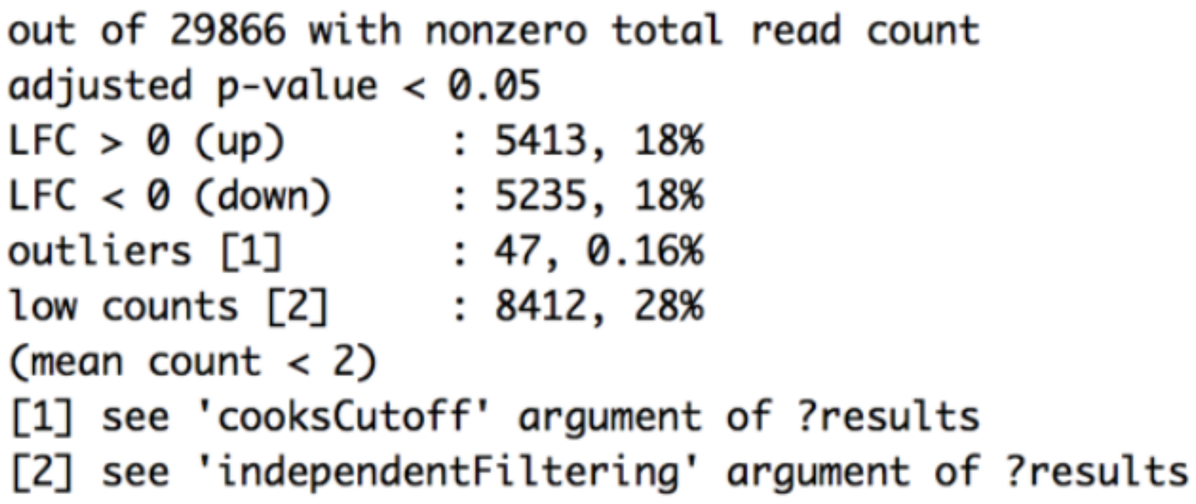
DESeq2 results table
to get descriptions for the columns in the results table
mcols(wt_res)
*to determine significant DE genes, we use the p-values adjusted for multiple test correction in the last column
head(wt_res, n=10)
ouput>



```
log2 fold change (MAP): condition fibrosis vs normal
Wald test p-value: condition fibrosis vs normal
data frame with 6 rows and 6 columns
                           baseMean   log2FoldChange            lfcSE             stat
                          <numeric>        <numeric>        <numeric>        <numeric>
ENSMUSG00000102693                0               NA               NA               NA
ENSMUSG00000064842                0               NA               NA               NA
ENSMUSG00000051951 19.5084656230804 3.55089043143673 0.648400500074659 4.66871842838828
ENSMUSG00000102851                0               NA               NA               NA
ENSMUSG00000103377                0               NA               NA               NA
ENSMUSG00000104017                0               NA               NA               NA
                                 pvalue                 padj
                              <numeric>            <numeric>
ENSMUSG00000102693                   NA                   NA
ENSMUSG00000064842                   NA                   NA
ENSMUSG00000051951 3.03084428526558e-06 1.93776447202312e-05
ENSMUSG00000102851                   NA                   NA
ENSMUSG00000103377                   NA                   NA
ENSMUSG00000104017                   NA                   NA
```

*remember what alpha 0.05 means, that for every gene tested has a 5% chance that the gene is called DE with it is not
when this happens, we have a false positive
put this into perspective, if we tested 50,000 genes we could have about 2,000 genes (5%) as false positives
because of this DESeq2 use multiple tes correction
uses the Benjamini-Hochberg method
this helps control the proportion of false positives relative to true positives
DESeq2 automatically filters out genes unlikely to be truly differentially expressed prior to testing
Which ones?:
-genes with 0 counts across all samples
-genes with low mean values across all samples
-genes with extreme count outliers
*above table, we can denote the filtered out genes by an 'NA'

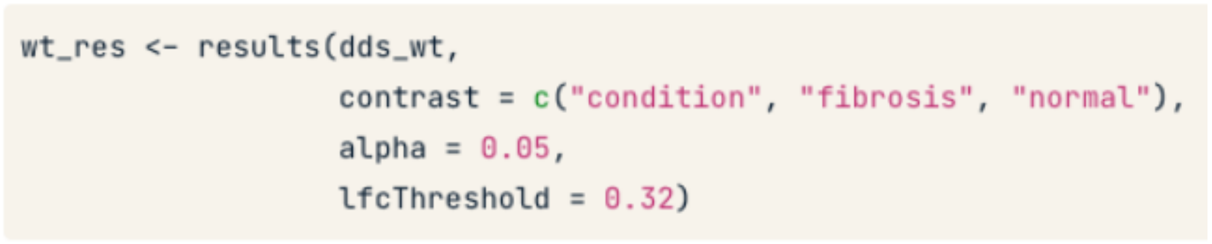
summary() function provides the number of DE genes for our alpha and info about the number of genes filtered



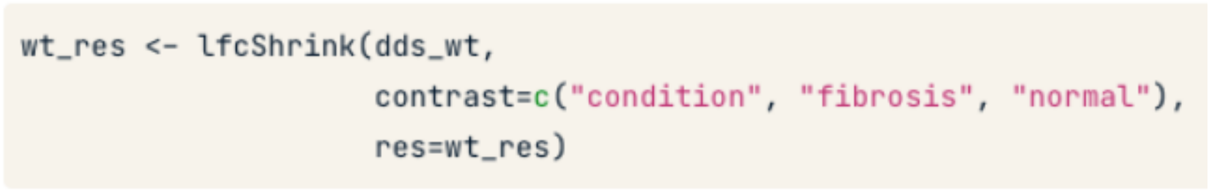
```
out of 29866 with nonzero total read count
adjusted p-value < 0.05
LFC > 0 (up)       : 5413, 18%
LFC < 0 (down)     : 5235, 18%
outliers [1]       : 47, 0.16%
low counts [2]     : 8412, 28%
(mean count < 2)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

our results give over 10,000 genes as DE
this is the sum of the DE genes with log2 fold changes less than 0 and greater than 0

This is a lot of genes
not ideal but sometimes necessary we can add a threshold



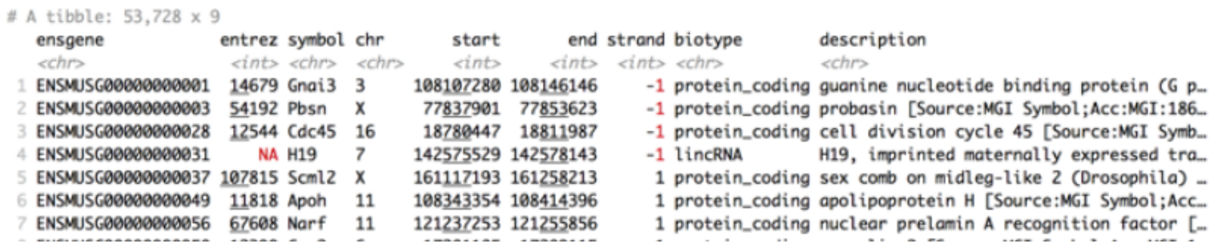
```
wt_res <- results(dds_wt,
                  contrast = c("condition", "fibrosis", "normal"),
                  alpha = 0.05,
                  lfcThreshold = 0.32)
```



```
wt_res <- lfcShrink(dds_wt,
                    contrast=c("condition", "fibrosis", "normal"),
                    res=wt_res)
```

here we use a 1.25 fold change threshold which equals 0.32 on the log2 scale
this increases the risk of losing biologically relevant genes
but by keeping the threshold low we hope to reduce the risk

To better understand which genes the results pertain to we use the annotables package
library(annotables)
example
grcm38 #known mouse genome



```
# A tibble: 53,728 x 9
  ensgene            entrez symbol chr       start       end strand biotype        description
  <chr>               <int> <chr>  <chr>     <int>     <int>  <int> <chr>          <chr>
1 ENSMUSG00000000001  14679 Gnai3  3     108107280 108146146     -1 protein_coding guanine nucleotide binding protein (G p…
2 ENSMUSG00000000003  54192 Pbsn   X      77837901  77853623     -1 protein_coding probasin [Source:MGI Symbol;Acc:MGI:186…
3 ENSMUSG00000000028  12544 Cdc45  16     18780447  18811987     -1 protein_coding cell division cycle 45 [Source:MGI Symb…
4 ENSMUSG00000000031     NA H19    7     142575529 142578143     -1 lincRNA        H19, imprinted maternally expressed tra…
5 ENSMUSG00000000037 107815 Scml2  X     161117193 161258213      1 protein_coding sex comb on midleg-like 2 (Drosophila) …
6 ENSMUSG00000000049  11818 Apoh   11    108343354 108414396      1 protein_coding apolipoprotein H [Source:MGI Symbol;Acc…
7 ENSMUSG00000000056  67608 Narf   11    121237253 121255856      1 protein_coding nuclear prelamin A recognition factor […
```

Next, extracting results

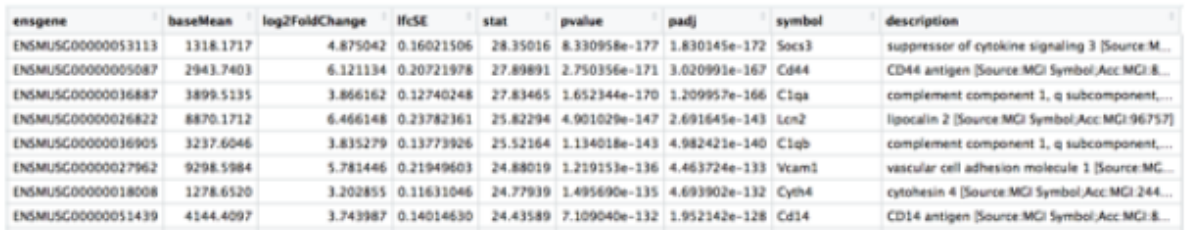
```
wt_res_all <- data.frame(wt_res) %>%
              rownames_to_column(var = "ensgene") %>%
              left_join(x = wt_res_all,
                        y = grcm38[, c("ensgene", "symbol", "description")],
                        by = "ensgene")
View(wt_res_all)
```

this annotates the genes with the gene names and descriptions
turn to a df
change the row names to a column
then merge the gene names and descriptions with our results using a left join and merge by ensemble gene IDs (ensgene)

Now extract the significant DE genes
subset our results table for genes with p-adjusted values less than 0.05

```
wt_res_sig <- subset(wt_res_all, padj < 0.05)
wt_res_sig  <- wt_res_sig %>%
        arrange(padj)
View(wt_res_all)
```

use arrange() to order the genes by p-adjusted values



ensgene	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj	symbol	description
ENSMUSG00000053113	1318.1717	4.875042	0.16021506	28.35016	8.330958e-177	1.830145e-172	Socs3	suppressor of cytokine signaling 3 [Source:M...
ENSMUSG00000005087	2943.7403	6.121134	0.20721978	27.89891	2.750356e-171	3.020991e-167	Cd44	CD44 antigen [Source:MGI Symbol;Acc:MGI:8...
ENSMUSG00000036887	3899.5135	3.866162	0.12740248	27.83465	1.652344e-170	1.209957e-166	C1qa	complement component 1, q subcomponent,...
ENSMUSG00000026822	8870.1712	6.466148	0.23782361	25.82294	4.901029e-147	2.691645e-143	Lcn2	lipocalin 2 [Source:MGI Symbol;Acc:MGI:96757]
ENSMUSG00000036905	3237.6046	3.835279	0.13773926	25.52164	1.134018e-143	4.982421e-140	C1qb	complement component 1, q subcomponent,...
ENSMUSG00000027962	9298.5984	5.781446	0.21949603	24.88019	1.219153e-136	4.463724e-133	Vcam1	vascular cell adhesion molecule 1 [Source:MG...
ENSMUSG00000018008	1278.6520	3.202855	0.11631046	24.77939	1.495690e-135	4.693902e-132	Cyth4	cytohesin 4 [Source:MGI Symbol;Acc:MGI:244...
ENSMUSG00000051439	4144.4097	3.743987	0.14014630	24.43589	7.109040e-132	1.952142e-128	Cd14	CD14 antigen [Source:MGI Symbol;Acc:MGI:8...

we should also see log2 foldchanges are >0.32
now we can explore this table for interesting or expected genes with a high

probability of being related to kidney fibrosis
lit review time

Example

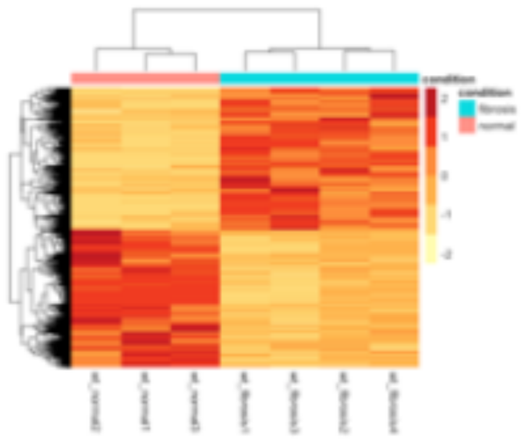
```
# Explore the results() function
?results

# Extract results
smoc2_res <- results(dds_smoc2,
            contrast = c("condition", "fibrosis", "normal"),
            alpha = 0.05,
            lfcThreshold = 0.32)

# Shrink the log2 fold changes
smoc2_res <- lfcShrink(dds_smoc2,
               contrast = c("condition", "fibrosis", "normal"),
               res = smoc2_res)

# Save results as a data frame
smoc2_res_all <- data.frame(smoc2_res)

# Subset the results to only return the significant genes with p-adjusted values
less than 0.05
smoc2_res_sig <- subset(smoc2_res_all, padj < 0.05)
```

Visualization of results
start with expression heatmap
explores the expression of significant genes
here we plot the normalized count values of the significant genes instead of the sample correlation values
subset the normalized counts to only the significant DE genes
example

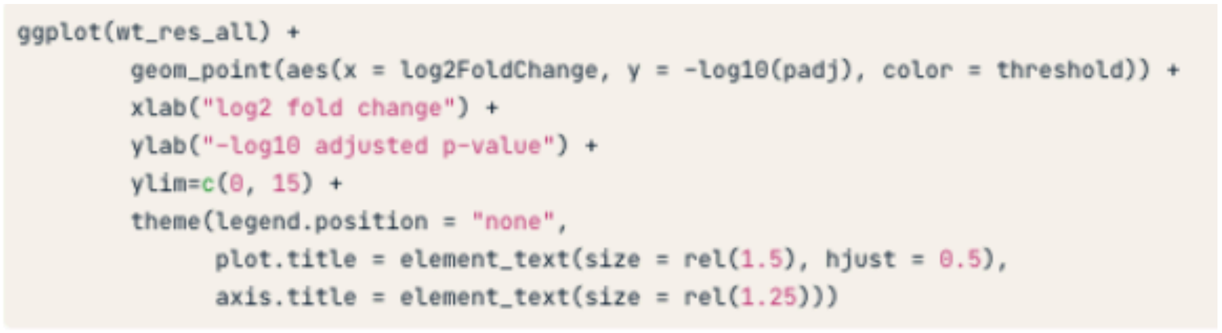
```
sig_norm_counts_wt <- normalized _counts_wt[wt_res_sig$ensgene, ]
#we can choose a color palette with RColorBrewer
library(RColorBrewer)
heat_colors <- brewer.pal(6, “YlOrRd”)
#to see all the possible color palettes
display.brewer.all()
```

```r
# Run pheatmap
pheatmap(sig_norm_counts_wt,
         color = heat_colors,
         cluster_rows = T,
         show_rownames = F,
         annotation = select(wt_metadata, condition),
         scale = "row")
```

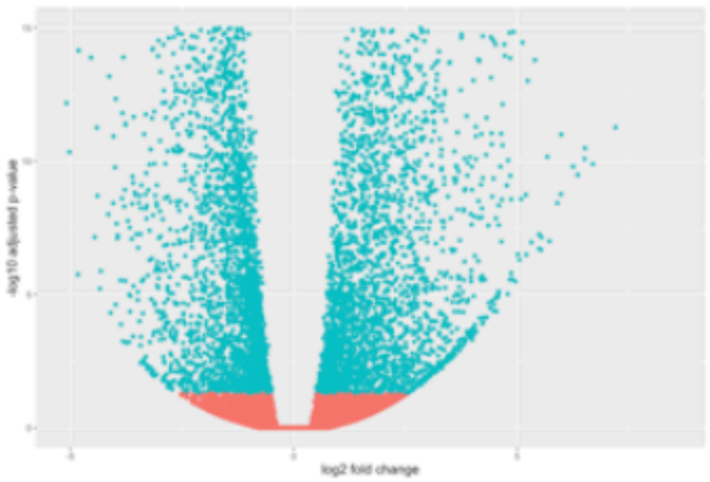
cluster by row, annotating with condition information and scaling by row
this plots Z-scores rather than the actual normalized count values
we expect to see the expression levels for the significant genes to cluster by sample group
output>



Additional useful plot - Volcano plot
shows the fold changes relative to the adjusted p-values for all genes



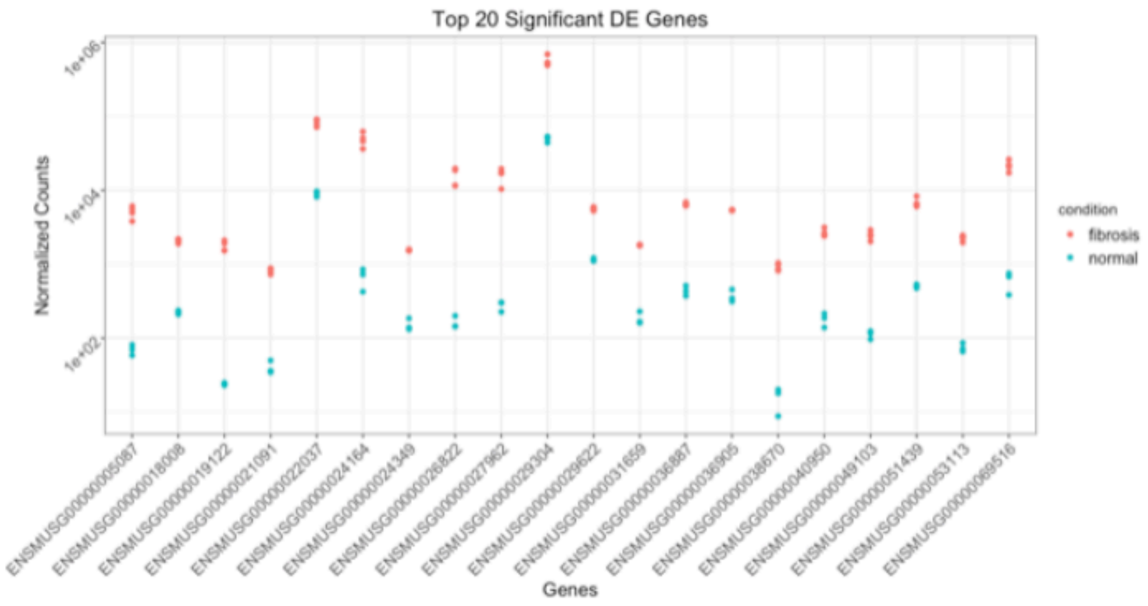
```
ggplot(wt_res_all) +
        geom_point(aes(x = log2FoldChange, y = -log10(padj), color = threshold)) +
        xlab("log2 fold change") +
        ylab("-log10 adjusted p-value") +
        ylim=c(0, 15) +
        theme(legend.position = "none",
              plot.title = element_text(size = rel(1.5), hjust = 0.5),
              axis.title = element_text(size = rel(1.25)))
```



ylim() allows us to better visualize the significance cut-off

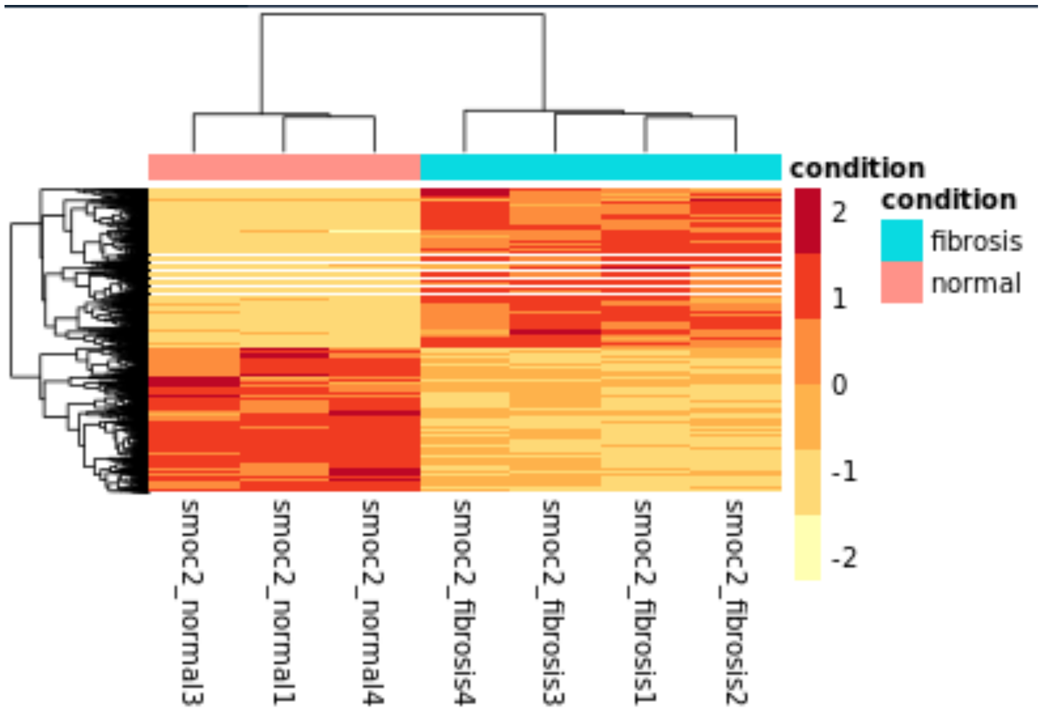
Visualizing the top significant genes
to do this create a dataframe
our example we want the top 20 genes
top_20 <- data.frame(sig_norm_counts_wt)[1:20,] %>%
    rownames_to_column(var = 'ensgene')
top_20 <- gather(top_20, key = "samplename", value = "normalized_counts", 2:8)
#2:8 is our last argument and specifies the columns we want to gather into a single column of values
to plot the expression we want to merge the metadata so that we can color the plot by sample group
we can use the inner_join() function to keep only those columns in both datasets
we join on the row names
we need to turn these into a column to join on them
we put the normalized counts on the y-axis
gene IDs on the x-axis
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))
#we use the log10 scale on the y-axis to more easily visualize the wide range in
expression values
```



Example

```
# Subset normalized counts to significant genes
sig_norm_counts_smoc2 <-
normalized_counts_smoc2[rownames(smoc2_res_sig), ]

# Choose heatmap color palette
heat_colors <- brewer.pal(n = 6, name = "YlOrRd")

# Plot heatmap
pheatmap(sig_norm_counts_smoc2,
        color = heat_colors,
        cluster_rows = TRUE,
        show_rownames = FALSE,
        annotation = select(smoc2_metadata, condition),
        scale = "row")

output>
```



DE Analysis Summary
From aligment or mapping
alignment or mapping of the reads to the genome to determine the location on the genome where the reads originated
reads derived from mRNA are often aligned to the organism's genome
some of these reads cross introns
tools for aligning reads to the genome need to align across introns or be splice-aware for RNA-seq
the ouput of alignment gives the genome coordinates for where the read most llikely originated from in the genome and info about the quality of the mapping
*the reads aligning to the exons of each gene are quantified to yield a matrix of gene counts
*the number of reads aligning to each of the genes is given in the count matrix and represents the expression level of the gene
*the more reads aligning to the gene, the higher the expression of the gene meaning more RNA transcripts were expressed
once we have the counts, differential expression analysis is performed
goal is to determine whether the gene counts between the sample groups are significantly different given the variation in the counts within the sample group

Example
# Check that all of the samples are in the same order in the metadata and count data

```
all(rownames(all_metadata) %in% colnames(all_rawcounts))

# DESeq object to test for the effect of fibrosis regardless of genotype
dds_all <- DESeqDataSetFromMatrix(countData = all_rawcounts,
                    colData = all_metadata,
                    design = ~ condition)

# DESeq object to test for the effect of genotype on the effect of fibrosis
dds_complex <- DESeqDataSetFromMatrix(countData = all_rawcounts,
                        colData = all_metadata,
                        design = ~ genotype + condition + genotype:condition)
```

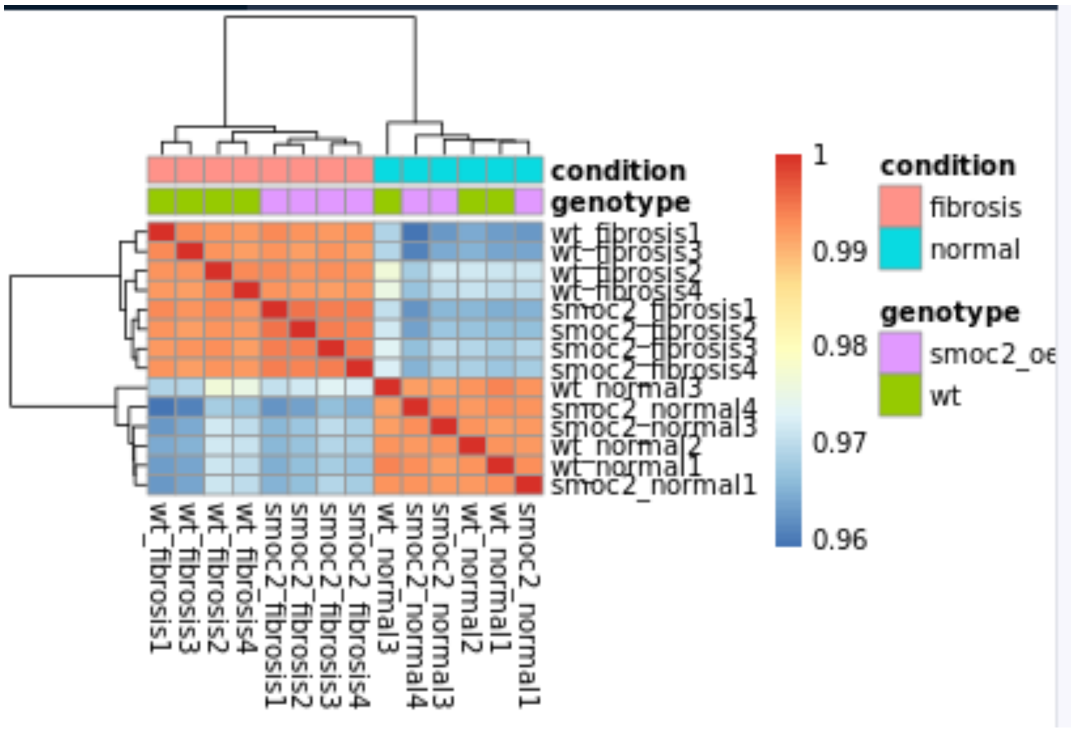
Recap of workflow - key notes
Normalization allows comparison of counts within a sample and between the different samples
to compare counts for the same gene bewteen different samples we need to normalize for library size, or total number of reads per sample
the normalized counts are needed to explore the similarity between samples using hierarchical correlation heatmaps and PCA
we log transform the normalized counts to improve the distances of the clusters for visualization
vst() function performs this transformation while being blind to sample group information
for the heatmap, we extract the transformed normalized counts using the assay() function on the transformed counts object
then compute the correlation values between samples using the cor() function

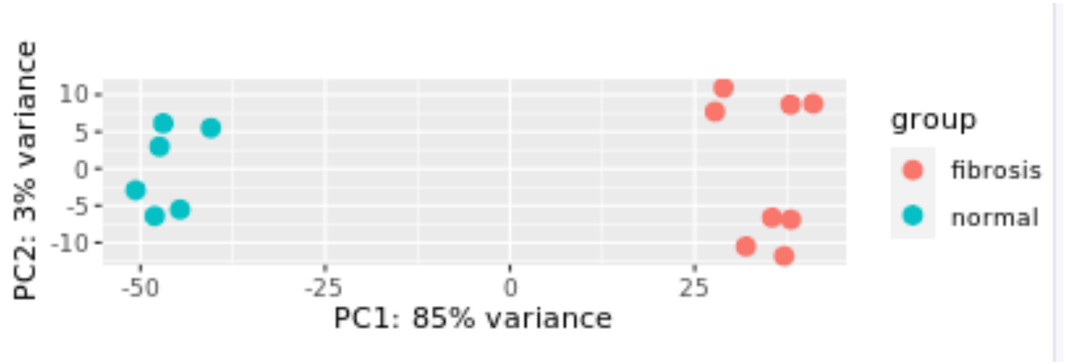
```
vsd %>%
    assay() %>% # Extract the vst matrix from the object
    cor() %>%   # Compute pairwise correlation values
    pheatmap(annotation = metadata[ , c("column_name1", "column_name2])
```

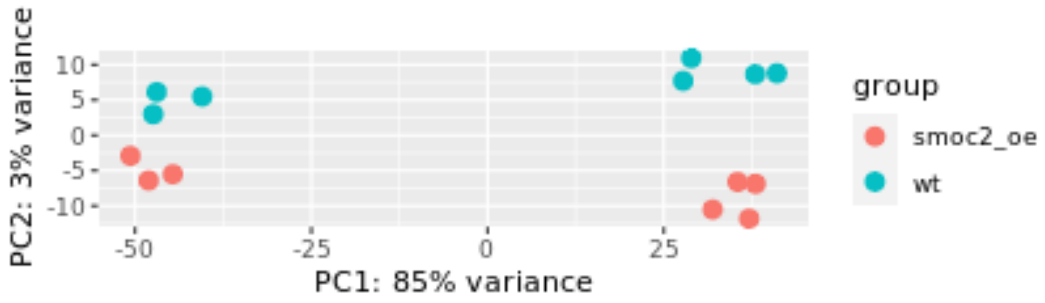
*remember we are looking for our biological replicates to cluster together and the sample groups to cluster apart
also identifies outliers which tend to have low correlation values with all other samples

Example
```
# Log transform counts for QC
vsd_all <- vst(dds_all, blind = TRUE)

# Create heatmap of sample correlation values
```

```
vsd_all %>%
        assay() %>%
        cor() %>%
        pheatmap(annotation = select(all_metadata, c("genotype", "condition")))

# Create the PCA plot for PC1 and PC2 and color by condition
plotPCA(vsd_all, intgroup = "condition")

# Create the PCA plot for PC1 and PC2 and color by genotype
plotPCA(vsd_all, intgroup = "genotype")

output>
```







```
# Select significant genese with padj < 0.05
smoc2_sig <- subset(res_all, padj < 0.05) %>%
            data.frame() %>%
            rownames_to_column(var = "geneID")

# Extract the top 6 genes with padj values
smoc2_sig %>%
   arrange(padj) %>%
   select(geneID, padj) %>%
   head(6)
```