# Intro To RNA-Seq Analysis

Serghei Mangul, Ph.D.
University of Southern California,
School of Pharmacy

# Insights from from RNA

From RNA seq data we can determine:

- Differentially expressed genes
- Cell type composition
- Tissue origin on the host's body
- Presence of microbes
- Clonality of immune cells

# Learning Objectives

**By the end of this module, students will be able to:**

1. Perform a global alignment of human reads to human reference genome using **Hisat2** and obtain gene counts using **htseq-count**
2. Obtain the gene transcripts directly from RNA Sequence Data using **Salmon.**
3. Perform a differential expression analysis using the **DESeq2** Bioconductor package in **R**.

# Key Terms

- **RNA Reads:**
  - Short (~100bp) stretches of RNA produced by sequencing machine
- **Gene Transcript:**
  - An RNA molecule of defined size over the length of a gene.
  - Multiple Gene Transcripts per gene
- **Alternative Splicing:**
  - Occurs by rearranging the pattern of intron and exon elements that are joined by **splicing** to alter the mRNA coding sequence.
  - A process that enables a messenger RNA to direct synthesis of different protein variants (isoforms) that may have different cellular functions or properties.

# Tools

- **Hisat2:**
  - An end-to-end alignment tool for mapping DNA and RNA to a population of human or microbial genomes as well as to a single reference genome.
  - Competitors: BWA, Hisat2, Blast
- **Samtools:**
  - Converts sam file to bam file format and generates index for conventional use with downstream bioinformatics pipelines.
- **Htseq-count:**
  - Counts how many reads map to each feature given a file with aligned sequencing reads and a list of genomic features,
- **Salmon:**
  - Quantifies the gene transcripts from RNA sequence data.
  - Competitors: Kallisto, Star
- **Deseq2:**
  - Performs a differential expression analysis by calculating geometric mean for each gene across all samples and eliminating outliers below a set threshold.

# Why 2 pipelines to do the same thing?

Like most scientific disciplines, Bioinformatics is a constantly evolving field. Accordingly, there are multiple ways to extract biological signals from genetic data. The best approach really depends on which tool is best suited for your needs.

# Pipeline 1

**RNA -> Hisat2 -> Samtools -> Htseq-count -> Gene counts -> gene matrix -> Deseq2 -> Visualize**

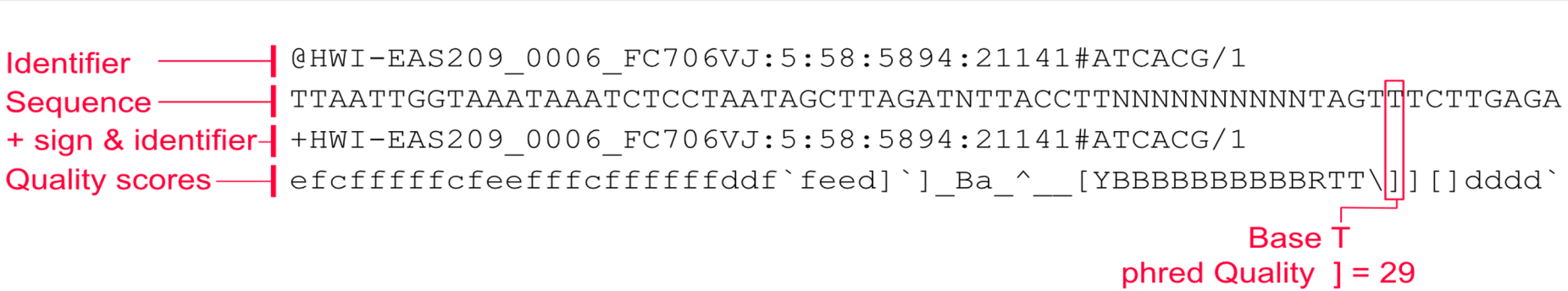
# Pipeline 1

**RNA -> Hisat2**

1. Align the RNA sequence data to a human reference genome, which generates a sam file containing the alignment.

# Inputs

## 1) Fastq file




## 2) Fasta File (Human DNA Reference Sequence)

>Human:Chromosome1
ATGCATCGGCTAGGTCTCTTGATCGATCGATGCTAGCTACGTACGTAC
>Human:Chromosome2
ATGCATCGGCTAGGTCTCTTGATCGATCGATGCTAGCTACGTACGTAC
>Human:Chromosome3
ATGCATCGGCTAGGTCTCTTGATCGATCGATGCTAGCTACGTACGTAC

# Commands

```
# generate Index from human DNA reference
!reference_seq="Homo_sapiens.GRCh38.dna.fa"


# Perform Alignment of RNA seq reads against reference
!hisat2 -x $reference_seq -1 simulated_reads/Sample1_control_R1.fq -2
simulated_reads/Sample1_control_R2.fq -S sam_files/Sample1_control.sam
```

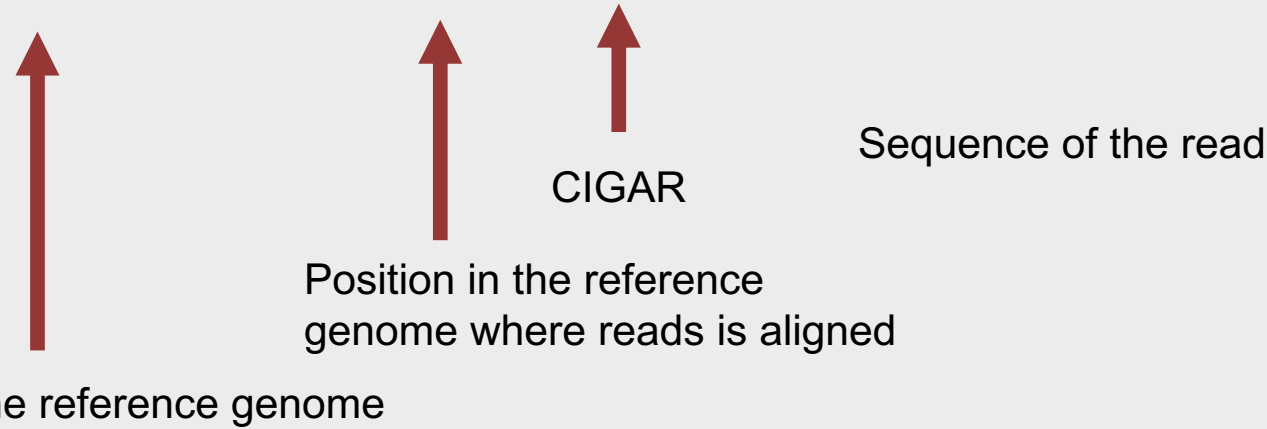
# Output: Sam File

```
Run this command to view aligned reads within the sam file:
~> less reads.sam

human_read1   0   human_reference   150    18M1I52M    MCCCTTTTAGTCAGTGTGGAAAAATCT..

human_read2   0   human_reference   1702   3M1D68M     CAGCCCACCAGAAGAGAGCGGCAGGTCT..

human_read3   0   human_reference   5902   48M1I46M    GTGGAAGGAAGCAACCACCACTCTATTT..
```



# Pipeline 1

## Hisat2 -> Samtools

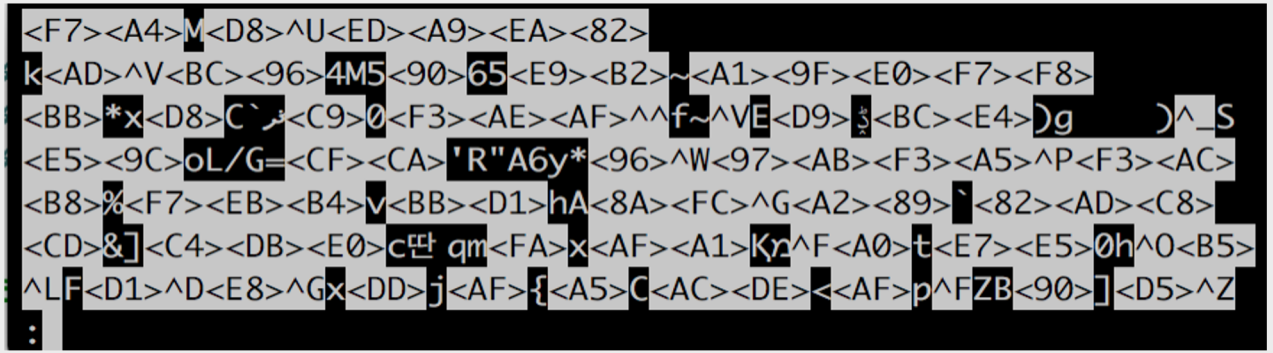
2. Convert the Sam file to a Bam file and sort the Bam file. This step is necessary because the Bam file is the binary version of a Sam file, which decreases the overall computational cost in downstream analysis of the alignment. Since we are working with Paired End reads, we sort the bam file by read name which is required by htseq-count.

# Input + Command

**Input:**

The input for converting to Bam is the sam file shown in the previous slide.

**Output:**
Binary/Compressed non-human-readable version of the sam file.




**Command:**

```
!samtools view -bS sam_files/Sample1_control.sam > bam_files/Sample1_control.bam
!samtools sort -n bam_files/Sample1_control.bam -o
```

# Pipeline 1

**Samtools -> Htseq-count**

3. Htseq-count quantifies gene counts by reading the conventional whole-genome alignment from the compressed and sorted bam file.

# Input + Command

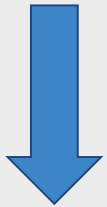
**Input:**

Bam file shown in previous slide + Human reference sequences for Gene Names

***gene transcripts reference is different from human genome reference**

Sorted Bam File

Human Gene Transcripts reference sequences: Gene transfer file (gtf)



**Command:**

```
!htseq-count -f bam bam_files/Sample1_control.sorted.bam htseq-count/Homo_sapiens.GRCh38.77.gtf
> htseq-count/Sample1.txt
```

Output file: captured transcripts and their counts

# Output:

Two columns containing the gene name and number of times that gene was expressed in the RNA seq data.

## Htseq-Count -> gene matrix

# Pipeline 1

1. Use Pandas dataframes to combine the gene name output files for each sample into one matrix.

## Input

Sample1.txt

```
ENSG00000000003 0
ENSG00000000005 0
ENSG00000000419 0
ENSG00000000457 0
ENSG00000000460 0
ENSG00000000938 0
ENSG00000000971 0
ENSG00000001036 0
ENSG00000001084 0
ENSG00000001167 0
ENSG00000001460 2
ENSG00000001461 0
ENSG00000001497 0
ENSG00000001561 0
ENSG00000001617 0
ENSG00000001626 0
ENSG00000001629 0
ENSG00000001630 0
ENSG00000001631 0
ENSG00000002016 0
ENSG00000002079 0
```

Sample3.txt

```
ENSG00000000003 0
ENSG00000000005 0
ENSG00000000419 0
ENSG00000000457 0
ENSG00000000460 0
ENSG00000000938 0
ENSG00000000971 0
ENSG00000001036 0
ENSG00000001084 0
ENSG00000001167 0
ENSG00000001460 0
ENSG00000001461 0
ENSG00000001497 0
ENSG00000001561 0
ENSG00000001617 0
ENSG00000001626 0
ENSG00000001629 0
ENSG00000001630 0
ENSG00000001631 0
ENSG00000002016 0
ENSG00000002079 0
ENSG00000002330 0
ENSG00000002549 0
ENSG00000002586 0
```

Sample2.txt

```
ENSG00000000003 0
ENSG00000000005 0
ENSG00000000419 0
ENSG00000000457 0
ENSG00000000460 0
ENSG00000000938 0
ENSG00000000971 0
ENSG00000001036 0
ENSG00000001084 0
ENSG00000001167 0
ENSG00000001460 1
ENSG00000001461 0
ENSG00000001497 0
ENSG00000001561 0
ENSG00000001617 0
ENSG00000001626 0
ENSG00000001629 0
ENSG00000001630 0
ENSG00000001631 0
ENSG00000002016 0
ENSG00000002079 0
ENSG00000002330 0
ENSG00000002549 0
ENSG00000002586 0
ENSG00000002587 0
ENSG00000002726 0
ENSG00000002745 0
ENSG00000002746 0
ENSG00000002822 0
ENSG00000002834 0
```

## Output

```
head(countData)

# count_matrix: should be a dataframe with each column as count,
#  and a id column for gene id
#          example:
#          id     sampleA     sampleB
#          geneA     5          1
#          geneB     4          5
#          geneC     1          2
```

A data.frame: 6 × 7

| | ENSEMBL_GeneID | Sample1 | Sample2 | Sample3 | Sample4 | Sample5 | Sample6 |
|---|---|---|---|---|---|---|---|
| | <fct> | <int> | <int> | <int> | <int> | <int> | <int> |
| 1 | ENSG00000000003.14 | 3 | 2 | 27 | 8 | 5 | 6 |
| 2 | ENSG00000000005.5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | ENSG00000000419.12 | 1057 | 1338 | 1453 | 1289 | 921 | 1332 |
| 4 | ENSG00000000457.13 | 463 | 499 | 561 | 386 | 532 | 542 |
| 5 | ENSG00000000460.16 | 634 | 418 | 1170 | 611 | 916 | 637 |
| 6 | ENSG00000000938.12 | 2616 | 3698 | 3110 | 2690 | 1897 | 1911 |

**Note:**

Now that we have generated the gene name matrix for all the samples in pipeline 1 we will generate the gene name matrix using a different pipeline. Then we will show how to generate the final differential analysis using Deseq2 starting from a gene matrix.

# Pipeline 2

**RNA -> Salmon -> transcripts -> gene matrix -> Deseq2 -> Visualize**

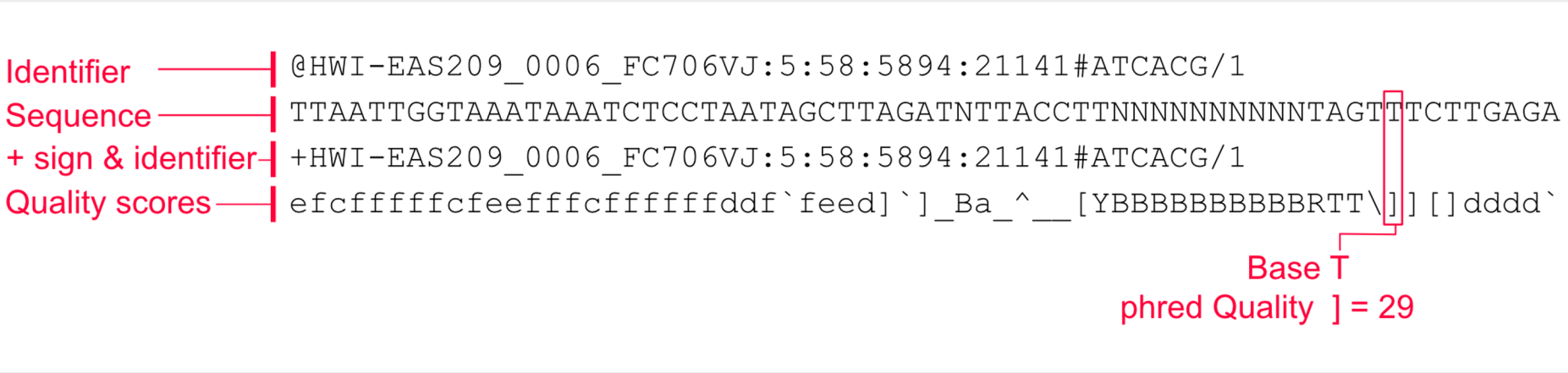
## RNA -> Salmon

# Pipeline 2

1. Salmon uses a kmer based approach to gene transcript identification and quantification. This bypasses the need for alignment and is more computationally efficient.

# Inputs

## 1) Fastq file




## 2) Human Gene Transcripts Reference File

>ENST00000632684.1 cdna chromosome:GRCh38:7:142786213:142786224:1 gene:ENSG00000282431.1 gene_biotype:TR_D_gene transcript_biotype:TR_D_gene gene_symbol:AC245427.8
GGGACAGGGGGC

>ENST00000434970.2 cdna chromosome:GRCh38:14:22439007:22439015:1 gene:ENSG00000237235.2 gene_biotype:TR_D_gene transcript_biotype:TR_D_gene gene_symbol:TRDD2 description:T-cell receptor delta diversity 2 [Source:HGNC Symbol;Acc:HGNC:12255]
CCTTCCTAC

# Command

```
# create salmon index out of ensemble gene transcripts for human GRCh38
!salmon-latest_linux_x86_64/bin/salmon index -t
salmon/Homo_sapiens.GRCh38.cdna.all.fa.gz -i salmon/transcripts_index


# run salmon quantification on reads


# Sample1
!salmon-latest_linux_x86_64/bin/salmon quant -i salmon/transcripts_index -l A -1
simulated_reads/Sample1_control_R1.fq  -2 simulated_reads/Sample1_control_R2.fq -p
8 --validateMappings -o salmon/sample1_transcripts_quant
```

# Output

**Quant.sf file:** Key columns are Gene transcript and NumReads

salmon.Sample1.quant.sf

| | Name | Length | EffectiveLength | TPM | NumReads |
|---|---|---|---|---|---|
| 2 | ENST00000632684.1 | 12 | 13.000 | 0.000000 | 0.000 |
| 3 | ENST00000434970.2 | 9 | 10.000 | 0.000000 | 0.000 |
| 4 | ENST00000448914.1 | 13 | 14.000 | 0.000000 | 0.000 |
| 5 | ENST00000415118.1 | 8 | 9.000 | 0.000000 | 0.000 |
| 6 | ENST00000390567.1 | 20 | 21.000 | 0.000000 | 0.000 |
| 7 | ENST00000439842.1 | 11 | 12.000 | 0.000000 | 0.000 |
| 8 | ENST00000454908.1 | 17 | 18.000 | 0.000000 | 0.000 |
| 9 | ENST00000390583.1 | 31 | 32.000 | 0.000000 | 0.000 |
| 10 | ENST00000390572.1 | 28 | 29.000 | 0.000000 | 0.000 |
| 11 | ENST00000390571.1 | 31 | 32.000 | 0.000000 | 0.000 |
| 12 | ENST00000454691.1 | 18 | 19.000 | 0.000000 | 0.000 |
| 13 | ENST00000390588.1 | 20 | 21.000 | 0.000000 | 0.000 |
| 14 | ENST00000390581.1 | 23 | 24.000 | 0.000000 | 0.000 |
| 15 | ENST00000390574.1 | 21 | 22.000 | 0.000000 | 0.000 |
| 16 | ENST00000450276.1 | 17 | 18.000 | 0.000000 | 0.000 |
| 17 | ENST00000431870.1 | 16 | 17.000 | 0.000000 | 0.000 |
| 18 | ENST00000414852.1 | 16 | 17.000 | 0.000000 | 0.000 |

## Gene Transcripts -> Gene Names matrix

# Pipeline 2

1. Use bioconductor R package to generate a matrix of gene counts per sample for preparing input for Deseq2.

*Unlike Htseq-count, Salmon extracts the gene transcripts (not the gene names). This means we need the extra step to sum up the gene transcripts that correspond to one gene name before we can construct the final gene matrix. There is a standard R package to achieve this, which requires a reference of gene transcripts to gene names.

**Note:**

Htseq Count outputs
Gene IDs:

ENSG0000022397.25

Salmon outputs
Transcript IDs:

ENST00000632684.1

This means we have to add an extra step in Salmon output (quant.sf files) and convert the gene transcripts to gene IDs. This is because the relationship between transcripts to genes is many to one. We do this using an R package which loads a reference library performs the sum and conversion and outputs a gene id matrix! **We will show and example of this in the Breakout Session.**

# Input + Command

**Input:** The quant.sf files generated from salmon that contain the gene transcripts and their counts for each sample.

## Command (R code):

```
samples <- read.table(file.path(dir, "samples.txt"), header = TRUE)

files <- file.path(dir, "salmon", samples$run, "quant.sf")
names(files) <- paste0("sample", 1:6)

library(TxDb.Hsapiens.UCSC.hg19.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
k <- keys(txdb, keytype = "TXNAME")
tx2gene <- select(txdb, k, "GENEID", "TXNAME")

library(readr)
tx2gene <- read_csv(file.path(dir, "tx2gene.gencode.v27.csv"))
head(tx2gene)
```

## Note:

Now that we have shown to different ways using bioinformatics tools to generate gene count matrices, we will proceed to generate the final differential analysis using the Deseq2 R package. The following process is the same for both pipelines because it only requires a gene matrix and metadata file.

## Gene matrix -> Deseq2

# Pipelines 1 & 2

Deseq2 performs a differential expression analysis using the gene counts and metadata.

Note: This must be done in R. We have created a separate notebook for you to load your gene matrix and run deseq2 in an R environment.

## Inputs for Deseq2:

1) **Combined Matrix containing the gene names and counts for each sample shown in previous slide.**

A data.frame: 6 × 7

| | ENSEMBL_GeneID | Sample1 | Sample2 | Sample3 | Sample4 | Sample5 | Sample6 |
|---|---|---|---|---|---|---|---|
| | <fct> | <int> | <int> | <int> | <int> | <int> | <int> |
| 1 | ENSG00000000003.14 | 3 | 2 | 27 | 8 | 5 | 6 |
| 2 | ENSG00000000005.5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | ENSG00000000419.12 | 1057 | 1338 | 1453 | 1289 | 921 | 1332 |
| 4 | ENSG00000000457.13 | 463 | 499 | 561 | 386 | 532 | 542 |
| 5 | ENSG00000000460.16 | 634 | 418 | 1170 | 611 | 916 | 637 |
| 6 | ENSG00000000938.12 | 2616 | 3698 | 3110 | 2690 | 1897 | 1911 |

1) **Metadata Dataframe**

A data.frame: 6 × 2

| | id | dex |
|---|---|---|
| | <fct> | <fct> |
| Sample1 | | control |
| Sample2 | | control |
| Sample3 | | control |
| Sample4 | | treated |
| Sample5 | | treated |
| Sample6 | | treated |

# Command:

## Construct DESEQDataSet Object:

```
dds <- DESeqDataSetFromMatrix(countData=countMatrix,
                              colData=metaData,
                              design=~dex, tidy = TRUE)
```

A data.frame: 6 × 7

| | ENSEMBL_GeneID | Sample1 | Sample2 | Sample3 | Sample4 | Sample5 | Sample6 |
|---|---|---|---|---|---|---|---|
| | <fct> | <int> | <int> | <int> | <int> | <int> | <int> |
| 1 | ENSG00000000003.14 | 3 | 2 | 27 | 8 | 5 | 6 |
| 2 | ENSG00000000005.5 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | ENSG00000000419.12 | 1057 | 1338 | 1453 | 1289 | 921 | 1332 |
| 4 | ENSG00000000457.13 | 463 | 499 | 561 | 386 | 532 | 542 |
| 5 | ENSG00000000460.16 | 634 | 418 | 1170 | 611 | 916 | 637 |
| 6 | ENSG00000000938.12 | 2616 | 3698 | 3110 | 2690 | 1897 | 1911 |

| id | dex |
|---|---|
| <fct> | <fct> |
| Sample1 | control |
| Sample2 | control |
| Sample3 | control |
| Sample4 | treated |
| Sample5 | treated |
| Sample6 | treated |

*Design parameter corresponds to the column which differentiates case vs control. In this case we called it dex

# Command + Output

**View the results of running Deseq2:** Sort results of differentially expressed genes by p-value

```
res <- results(dds)
res <- res[order(res$padj),]
```

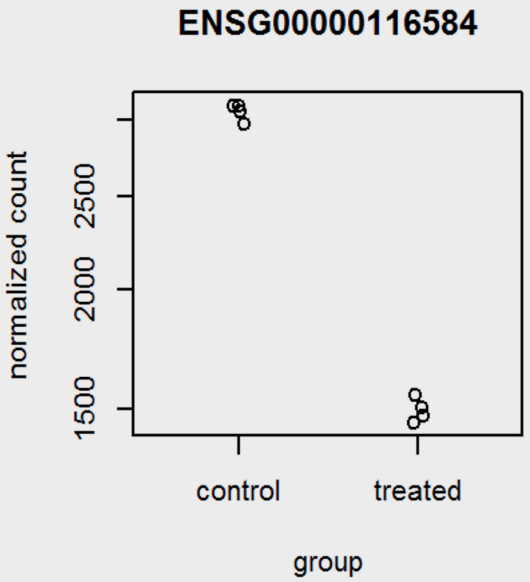
| GENEID | baseMean | log2FoldChange | lfcSE | stat | pvalue | padj |
|---|---|---|---|---|---|---|
| | <numeric> | <numeric> | <numeric> | <numeric> | <numeric> | <numeric> |
| ENSG00000152583 | 954.7709 | 3.966942 | 0.21424708 | 18.51573 | 1.541835e-76 | 2.332488e-72 |
| ENSG00000179094 | 743.2527 | 2.713956 | 0.16595960 | 16.35311 | 4.133115e-60 | 3.126288e-56 |
| ENSG00000116584 | 2277.9135 | -1.026994 | 0.06416315 | -16.00599 | 1.160615e-57 | 5.852596e-54 |
| ENSG00000189221 | 2383.7537 | 3.090702 | 0.19641099 | 15.73589 | 8.583635e-56 | 3.246331e-52 |
| ENSG00000120129 | 3440.7038 | 2.759321 | 0.18957216 | 14.55552 | 5.387437e-48 | 1.630023e-44 |
| ENSG00000148175 | 13493.9204 | 1.401941 | 0.09837916 | 14.25038 | 4.458290e-46 | 1.124084e-42 |

# Dot Plot of gene counts for top 3 differentially expressed genes

```
par(mfrow=c(2,3))

plotCounts(dds, gene="ENSG00000152583", intgroup="dex")
plotCounts(dds, gene="ENSG00000179094", intgroup="dex")
plotCounts(dds, gene="ENSG00000116584", intgroup="dex")
```

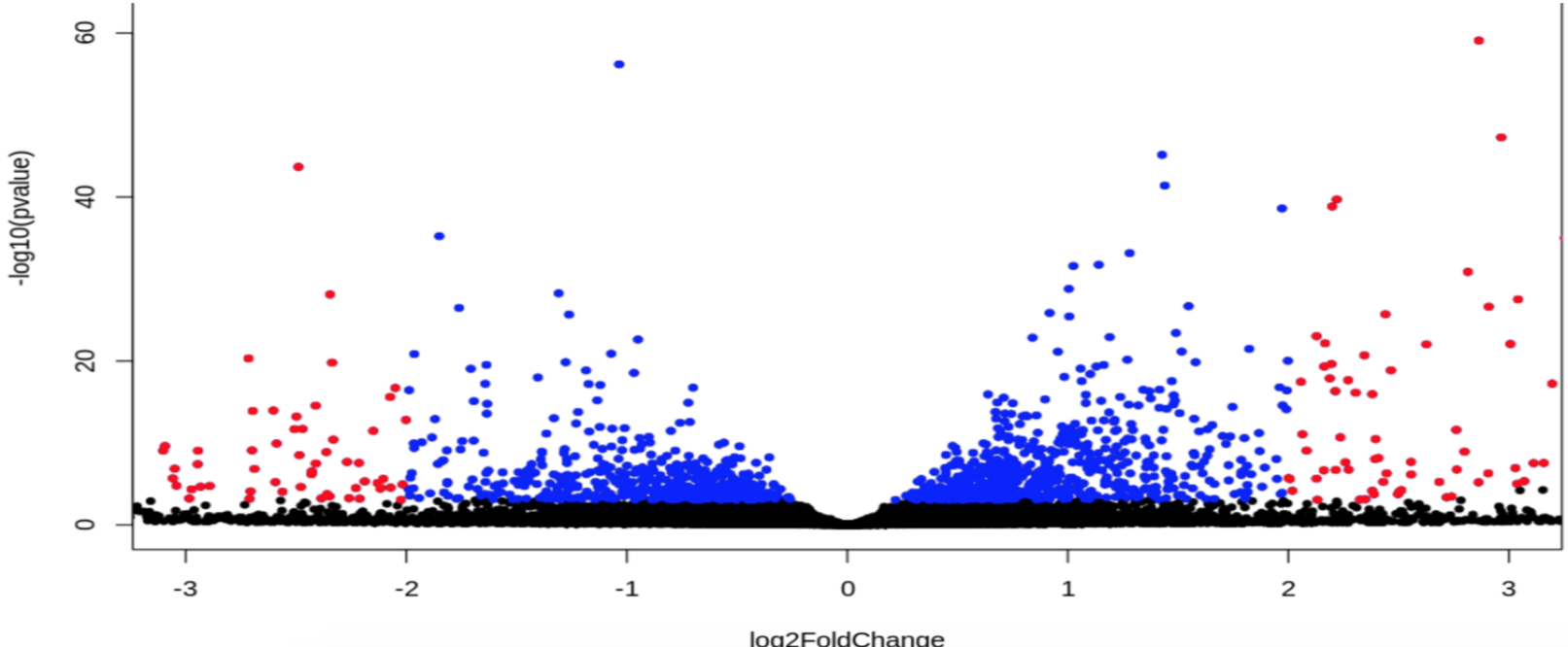




# Volcano Plot

```
#reset par
par(mfrow=c(1,1))
# Make a basic volcano plot
with(res, plot(log2FoldChange, -log10(pvalue), pch=20, main="Volcano plot", xlim=c(-3,3)))

# Add colored points: blue if padj<0.01, red if log2FC>1 and padj<0.05)
with(subset(res, padj<.01 ), points(log2FoldChange, -log10(pvalue), pch=20, col="blue"))
with(subset(res, padj<.01 & abs(log2FoldChange)>2), points(log2FoldChange, -log10(pvalue), pch=20, col="red"))
```



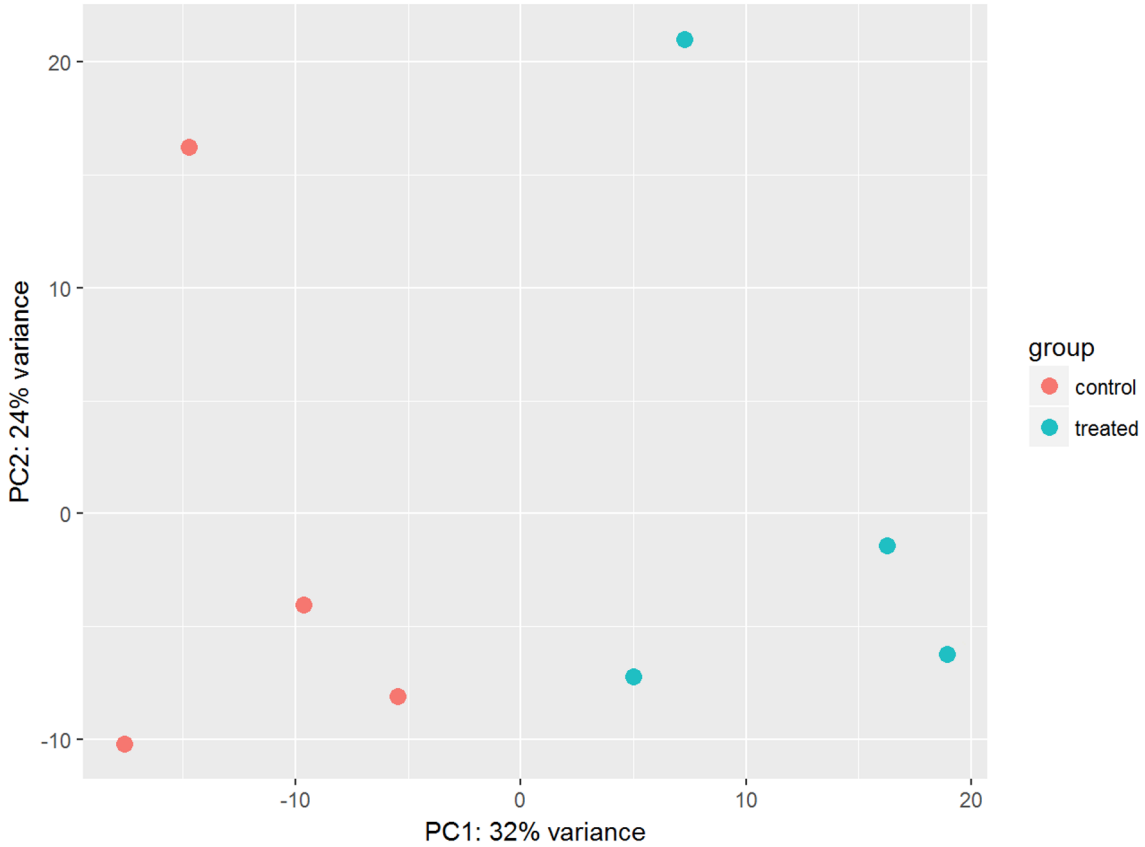
# Plot Variance Stabilizing Transformation data to show how samples group by treatment

1) R function vst() will perform the variance stabilizing transformation calculation. Save to vsData variable.

```
vsdata <- vst(dds, blind=FALSE)
```

1) Plot the transformed data based on case vs control.

```
plotPCA(vsdata, intgroup="dex")
```



# RNA-Sequence Analysis Workflow

- 1. Quality assess and clean raw sequencing data
- 2. Align reads to a reference
- 3. Count the number of reads assigned to each contig/gene
- 4. Extract counts and store in a matrix
- 5. Create column metadata table
- 6. Analyze count data using DESEQ2