**OVERALL WORKFLOW**



**TEXT in GREEN is a specific argument for each function/parameter**

**Download sra for ubuntu and extract the contents of the tar file:**

wget -output-document SRAtoolkit.tar.gz <https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-ubuntu64.tar.gz>

gunzip \*.tar.gz

tar -vxzf sratoolkit.tar

##### Append the path to the binaries to your PATH environment variable:

export PATH=$PATH:$PWD/sratoolkit.3.0.0-mac64/bin

##### Verify that the binaries will be found by the shell:

##### which fastq-dump

##### Download fastq files and unzip contents. Reads 1 and 2 correspond to 5’ and 3’ ends of the same cDNA fragment

##### ls

##### srrxxxx\_fastq.gz1 srrxxxx\_fastq.gz2

##### gunzip \*.fastq.gz#unzips the files

##### wc -l \*.gz #gives the length of all files

##### head \*. fastq

##### 

##### Files have sequence and a key for quality



*First we need to inspect for quality*

**Install “fastqc” binary**

sudo apt-get update

sudo apt-get install fastqc

**Perform Quality Control using fastqc**

fastqc <my\_file.ext> -o <output\_folder> time #time reports back how long it tool



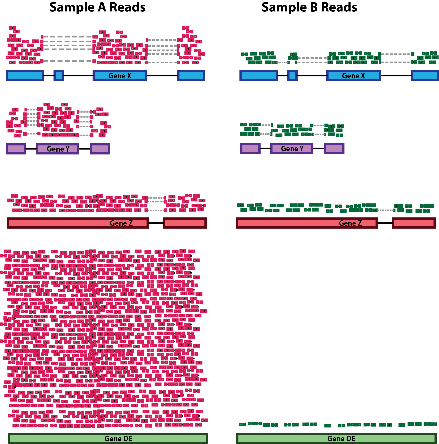
*Next we can trim off adapters and sequence we don’t need with “porechop”*

**Install “Porechop” binary**

sudo apt-get install Porechop

**Trim the sequences using “Porechop”**

Porechop my\_fastq\_file > trimmed\_output\_filename



Whichever method you choose Sequencing depth normalization is a crucial step in RNA-seq data analysis. Sequencing Depth: Sequencing depth refers to the total number of reads or fragments obtained from an RNA-seq experiment variability: Different samples may have varying sequencing depths due to technical or experimental reasons. When comparing gene expression levels between samples, we need to account for this variability. Poisson Distributions and Negative Binomial Distributions are both algorithmic that are used under the hood

R has 3 main ways to combat this in DESEQ2

1)Estimate Size factor and norm Transform

2) Variance stabilizing transformation

3) Rlog

**ALIGNMENT METHOD 1**



*We ready to align but first well need an indexed reference genome and convert to a binary format (BAM). We can index with “Bowtie” and align our reads with “bwa” and convert these human readable outputs to binary using samtools. This will allow us to Compare differential expression of our data to things like a GTF or GFF files for a given genome downstream. (GTF files contain intron exon positions)*

**Install “bowtie” and bwa binaries**

sudo apt get install bowtie

sudo apt get install bwa

**Index a genome using “bwa”**

bwa index -p genome <my\_genome\_file.fa>

**Align trimmed reads to indexed Genome**

bwa mem -t 2 trimmed\_output\_filename > output\_name.sam

**Convert human readable “sam” format to Binary “bam” format**

sudo apt install samtools

samtools view -S -b output\_name.sam > my\_alignment.bam

samtools sort -O <my\_sort.bam> <my\_alignment.bam>

**Obtain and decompress a “gtf” files**

*If you’re looking for information on gene structure or RNA expressions for a genome gtf.gz files contain this*

gunzip Sarrchoromyces\_cerevisiaw.R64-1-1.107.gtf.gz

**Mark or index the sorted binary file**

samtools index <my\_sort.bam>

*From here we can perform Differential expression using our annotated genome sequence binary which includes our reads and our unpacked GTF or GFF file using htseq*

**Install htseq**

sudo apt get install python.htseq

**check if the sam file is sorted using samtools**

samtools view -f 4 my\_alignment.sam

**Count the reads using the gtseq-count and save to a file**

**htseq-count -–f bam** <my\_sort.bam> | Sarrchoromyces\_cerevisiaw.R64-1-1.107.gtf > my\_output.ham

**ALIGNMENT METHOD 2**



*Another option for alignment is using a STAR aligner. This is a very rapid method details can be found at*

<https://github.com/alexdobin/STAR>

<https://github.com/alexdobin/STAR/blob/master/doc/STARmanual.pdf>

*The star method allows us to download a premade annotated genome index and generate an annotated genome upfront using a “genomeGenerate” run mode (--generateGenome option) and then align reads after. To do so we need to download FASTA.fa files of our reference genome (pick the chromosomes you need),* ***AND*** *a .GTF file so the software knows where the exon-exon junctions and where the introns are. Download what you need from EMSEMBLE.*

*Note\*\*\*(it is a good practice to keep all file types in separate directories of a project folder i.e. fastq, fastqc, GTF)*

**Get latest STAR source from releases**

wget https://github.com/alexdobin/STAR/archive/2.7.11b.tar.gz

tar -xzf 2.7.11b.tar.gz

cd STAR-2.7.11b

**Compile under Linux**

cd STAR/source

make STAR

**Compile under Mac OS X**

brew install gcc

**Build STAR: run 'make' in the source directory note that the path to c++ executable has to be adjusted to its current version**

cd source

make STARforMacStatic CXX=/usr/local/Cellar/gcc/8.2.0/bin/g++-8

**Make it available through the terminal**

cp STAR /usr/local/bin

**Download fasta (.fa) and exon\_intron (.gtf) files from ENSEMBLE and create an annotated genome**

*Unzip and Place each in a respective directory*

wgethttps://ftp.ensembl.org/pub/release-79/fasta/homo\_sapiens/dna/Homo\_sapiens.GRCh38.dna.chromosome.1.fa.gz

wget https://ftp.ensembl.org/pub/release-79/gtf/homo\_sapiens/Homo\_sapiens.GRCh38.79.gtf.gz

STAR –runmode genomeGenerate \

--genomeDIR directory\_to\_place\_new\_genome \ *#why is this even necessary??????*

--genomeFastaFiles path\_to\_genome-fasta\_file/ **Homo\_sapiens.GRCh38.dna.chromosome.1.fa** \

--sjdbGTFfile path\_to\_genome-GTF\_file/ **Homo\_sapiens.GRCh38.79.gtf\**

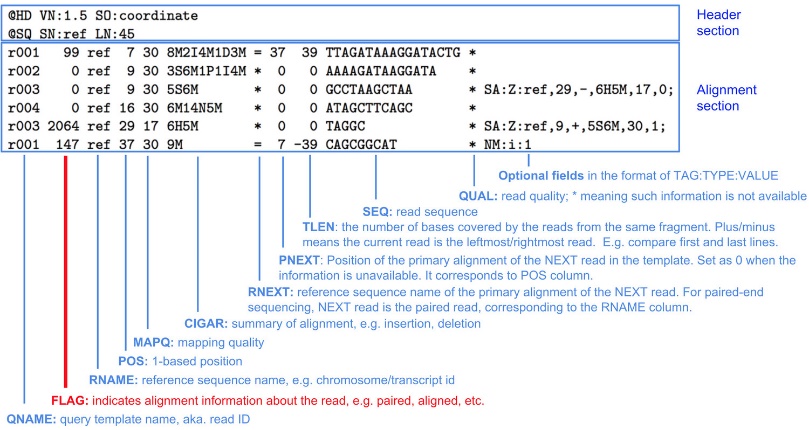
**--**readFilesIn path\_to\_first\_read/***srrxxxx\_1.fastq*** path\_to\_second\_read /***srrxxxx\_2.fastq***

**Using STAR on downloaded genome that has been annotated against the paired end reads**

STAR --runTHREAD 12 \ *# number of cores*

--genomeDIR directory\_containing\_newly\_annotated\_genome \

**--**readFilesIn directory\_containing\_reads/***srrxxxx\_1.fastq*** directory\_containing\_reads/***srrxxxx\_2.fastq***



**Ways to decipher SAM files can be found below**

<https://samtools.github.io/hts-specs/SAMv1.pdf>

<https://broadinstitute.github.io/picard/explain-flags.html>

**ALIGNMENT METHOD 3**



*IGV is another way to visualize BAM files you’ve created against a genome. You can upload and compare to premade index or upload your own indexed genome.*

*\*\*Click collapse to zoom out and see the total coverage for all genes across the genome*

*\*\*upload the BAM file (and index file) by clicking****[Tracks]****,*

<https://www.broadinstitute.org/igv>

**ALIGNMENT METHOD 4**



*Yet another method is to align reads to a Transcriptome. This is a translated version of the Genome (exons only). The expression of the Reads are expressed probabilistically*

<https://deweylab.github.io/RSEM/README.html>

<https://deweylab.github.io/RSEM/rsem-calculate-expression.html>

**Download and decompress the human genome and GTF files:**

ftp://ftp.ensembl.org/pub/release-83/fasta/homo\_sapiens/dna/Homo\_sapiens.GRCh38.dna.primary\_assembly.fa.gz

ftp://ftp.ensembl.org/pub/release-83/gtf/homo\_sapiens/Homo\_sapiens.GRCh38.83.gtf.gz

**build RSEM references:**

**\*\* notes several fasta files will be created**

rsem-prepare-reference --gtf Homo\_sapiens.GRCh38.83.gtf \ *#exon intron mapping*

genome Homo\_sapiens.GRCh38.dna.chromosome.1.fa \

rsemGenome/GRch38.79.chrom1 *# output directory and a new file name*

**Estimate gene isoform expression following RSEM reference build:**

**\*\* notes several fasta files will be created**

rsem-calculate expression -p 12 \ #threads

--paired end fastq/SRRxxxxxx\_1.fastq fastq/SRRxxxxxx\_2.fastq \ *#path to reads*

rsemGenome/GRch38.79.chrom1 SRRxxxxxxx *#output directory/ file name plus the*

*sample name*

**ALIGNMENT METHOD 5**



*Yet another method is from the BROAD is GATK or Genomic analysis toolkit. It can detect structural variance of different kinds using different algorithms*

[Getting started with GATK4 – GATK (broadinstitute.org)](https://gatk.broadinstitute.org/hc/en-us/articles/360036194592-Getting-started-with-GATK4)



The same workflow can also be down in R

But uses a slightly different syntax see also “An entire Sequence Pipeline.Rmd”



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Read in Bam Files GTF Files and Genome** | **Create txDB object of the GTF file** | **Create GRanges list**  **object** | **Perform alignment on Granges object and reads from samtools** | **Summarized Experiment object contains a count table** |
| **“RSAMTOOLS”**  **Converts bam files** | **“Genomic Features” makes a database object** | **“Genomic Features” makes a database object** | **“Genomics Alignment”**  **Creates a summarized Experiment object** | **“Summarized Experiment”** |



BioBase is another method

It creates an “Expressionset” object along with other meta data from raw Microarray data. This objects get passed down the pipeline as newly formatted objects

First Create a MIAME object using the function “new” with basic descriptions about the experiment (e.g., the investigator or lab where the experiment was done, an overall title, and other notes

experimentData <- new ("MIAME",

name="Pierre Fermat",

lab="Francis Galton Lab",

contact="pfermat@lab.not.exist", +

title="Smoking-Cancer Experiment",

abstract="An example ExpressionSet", +

url="www.lab.not.exist",

other=list (+ notes="Created from text files" +))

**Assembling an ExpressionSet object is created by assembling its component parts and calling ExpressionSet()**

exampleSet <- ExpressionSet(assayData=exprs,

phenoData=phenoData,

experimentData=experimentData,

annotation="hgu95av2”) +

**Expression sets can also be imported**

count.matrix <- exprs (wang. eset)[,10:21]

col.data <- pData(wang.eset)[10:21,]

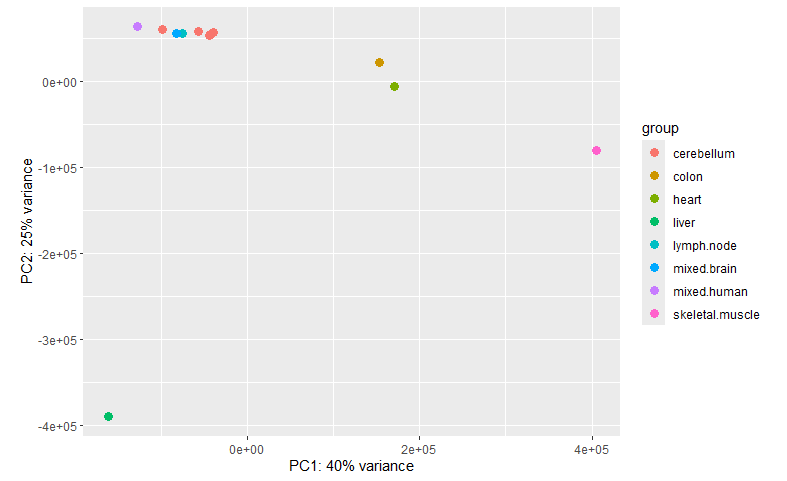
dds <- DESeqDataSetFromMatrix (count. matrix, col.data, design=~cell.type)

DESeqSummarizedExperiment(dds)

dds<- DESeqTransform(dds) #normilized read depth

**PCA Plot**

plotPCA (dds, Intgroup = "cell. Type")



**Converting to a result object with DESEQ *(not DESEQ2) allows us to compare a filter to counts across groups***

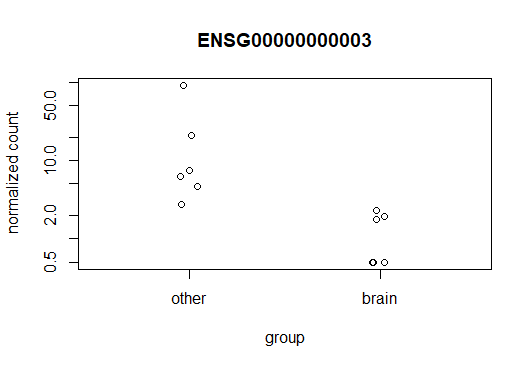
dds <- DESeq(dds)

res <- results(dds)

head(res)

table(res$padj < 0.1)

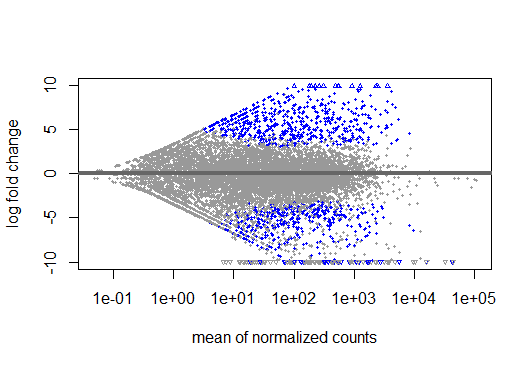
plotCounts(dds, gene=which.min(res$padj < 1), intgroup="type")



**The results object can also be filtered by Log2Fold change for more stringency**

res.thr <- results(dds, lfcThreshold=2)

plotMA(res.thr, ylim=c(-10,10))



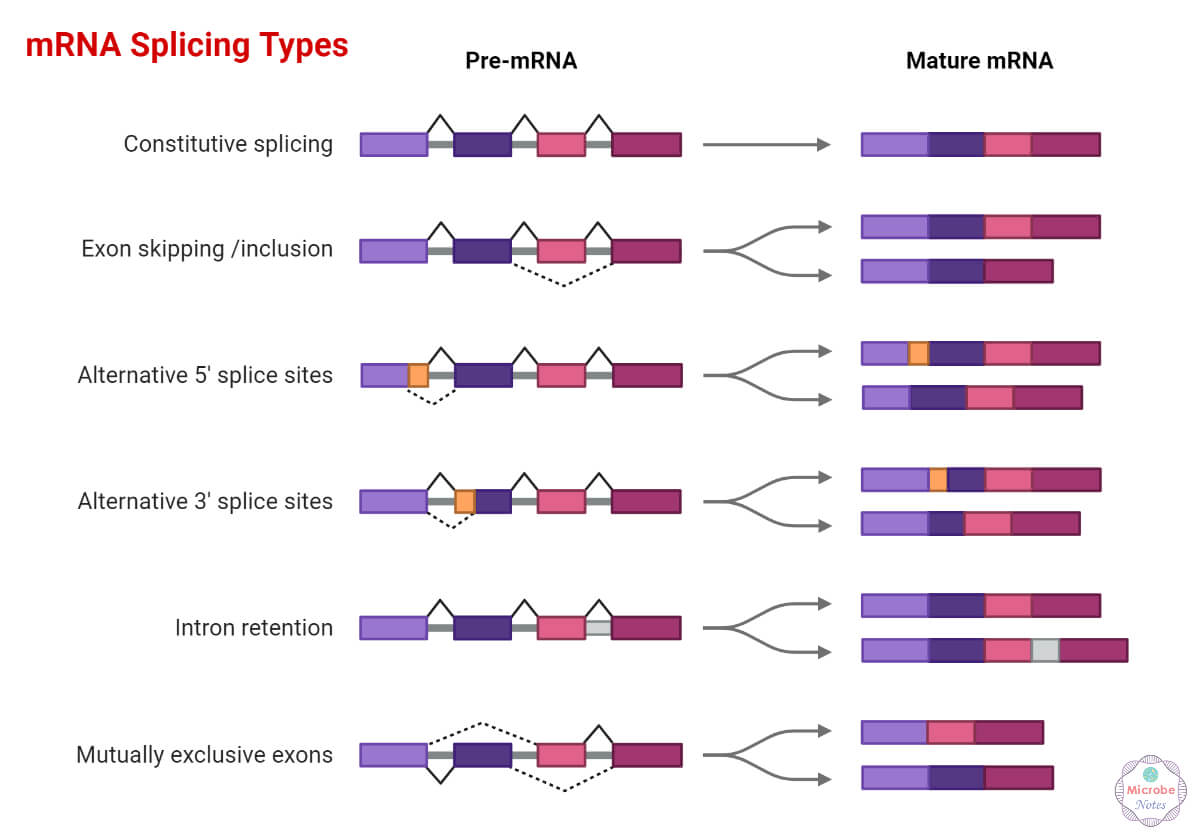
*Reads in blue fall within log fold change value*

*Lower count reads have been normalized*

**You can easily subset objects with assay**

assay(dds)[list,] #list by index number or name

**EXON INCLUSION**



*In addition, Exon usage or splicing can be affected with treatment or*

*disease corresponding to different isoforms of the same gene for this*

*the package “DEXSEQ” can analyze these differences*

**Build a DEXSeq dataset object**

library("pasilla")

inDir <- system.file("extdata", package="pasilla", mustWork=TRUE)

countFiles <- list.files(inDir, pattern="fb.txt$", full.names=TRUE)

flattenedFile <- list.files(inDir, pattern="gff$", full.names=TRUE)

sampleTable <- data.frame(

row.names = c("treated1", "treated2", "treated3",

"untreated1", "untreated2", "untreated3", "untreated4"),

condition = c("knockdown", "knockdown", "knockdown",

"control", "control", "control", "control")

)

library("DEXSeq")

dxd <- DEXSeqDataSetFromHTSeq(countFiles, sampleData=sampleTable,

design= ~ sample + exon + condition:exon,

flattenedfile=flattenedFile)

dxd <- estimateSizeFactors( dxd )

dxd <- estimateDispersions( dxd )

dxd <- testForDEU( dxd )

dxd <- estimateExonFoldChanges( dxd )

dxr <- DEXSeqResults( dxd )

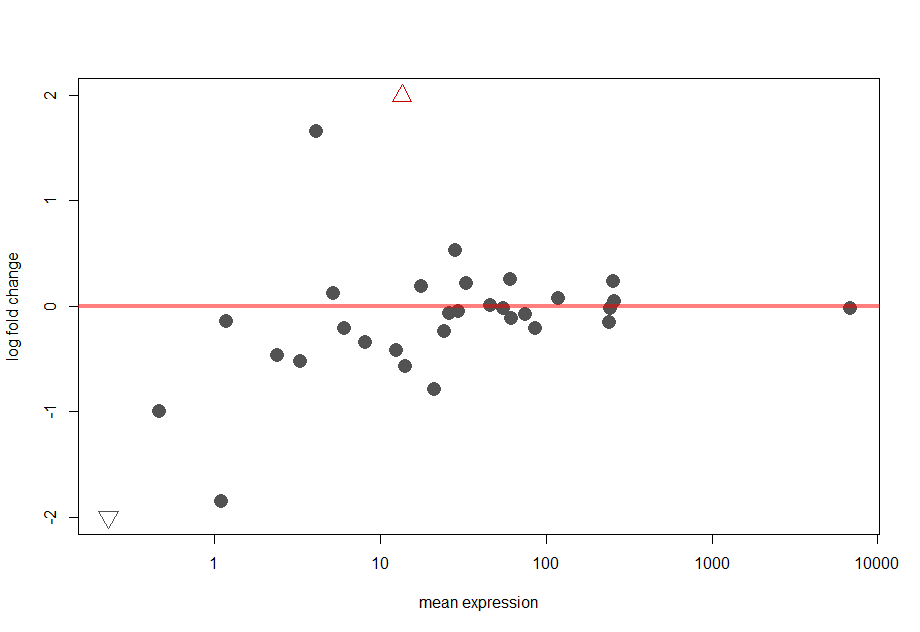
plotMA(dsr[which(dsr$featureID == "E009"),] ,

foldChangeColumn = "log2fold\_knockdown\_control",

norCounts=TRUE,

displayTranscripts=TRUE,

cex = 2)



plotDEXSeq( dsr, "FBgn0000256",norCounts=TRUE, displayTranscripts=TRUE )

