ANGSD 2020 Final Project

Explores the Differences in Gene Expression in
Human Lung Adenocarcinoma Samples
With Brain Metastasis

Saloni Vishwkarma

4/2/2020

Contents

Introduction

Brain metastases are the most common tumors of the central nervous system (CNS) and 40-50% of these cerebral metastases arise from primary lung tumors (Rute et al., 2018). Therefore, CNS metastasis is an emerging area of interest in organ-specific metastasis research. Lung and breast cancers are the most common types of primary tumors to develop brain metastases, which I predict is because of the proximity of these cancers to lymph nodes in the human body to easily facilitate spread via the lymphatic system (Gril et al., 2010). However, I am very curious to explore whether or not any genetic differences exist in cancers that metastasize to the brain. Brain metastases from lung cancer can occur the moment of or months after diagnosis of primary lung cancer (Rute et al., 2018) and are often detected during times of relapse (Hubbs et al., 2010). Thus, the prognosis remains bleak and therefore, prevention of brain metastasis is a critical concern in order to improve survival among lung cancer patients.

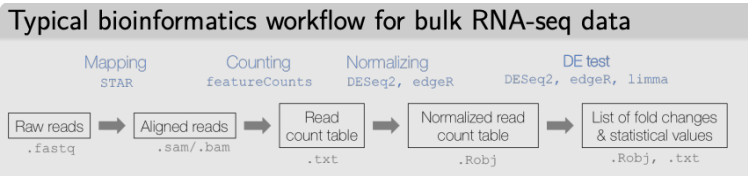
This research project conducted by Dong X analyses the RNA-Seq to examine primary lung tumors from NSCLC patients with and without BM and identify differentially expressed genes and potential signaling pathways. There is a total of 6 samples: 3 patients with brain metastasis (BM+) and other 3 without brain metastasis (BM-). All human cancer tissues used in this study were taken from surgical specimens and preserved immediately in liquid nitrogen after surgery. The scientists were able to identify a total of 566 differentially expressed genes between BM+ and BM- samples, so I'm eager to see whether I can replicate the data analysis pipeline to pinpoint the genes that play an important role in brain metastasis.

Main Question and Hypothesis

In my final project for ANGSD, I would like to explore the following question: What is the difference in genetic profiles between lung cancers that metastasize to the brain vs those that do not? I hypothesize that there will be a difference, especially in genes that are involved in cancer development (tumor supressors, apoptosis-regulators etc.) along with master regulator genes.

Methods and Results

As I outlined my data analysis, I refered to this flow chart to better understand the typical bioinformatics workflow:



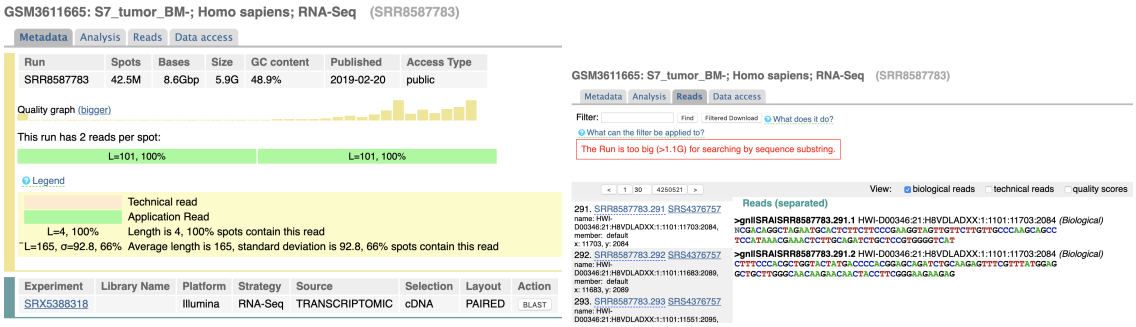
Data First Glance and Download

First and foremost, I logged onto aristotle, launched curie, and then into buddy/farina (using 50G of memory because the server kept kicking me out and timing me out if I used anything smaller). There, I made a new directory "finalproject" to store my downloaded data. Once, I found the paper I was interested in, I looked at the Gene Expression Omnibus (GEO Databases) to gain access to the data files. At https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE126548, I navigating through the 6 samples links and found the Sequence Read Archive (SRA) download links. I downloaded them individually using the wget command. If I had more samples or more runs, I would have implemented a for loop.

```
> ssh aristotle.med.cornell.edu -l spv4001
> ssh curie.pbtech
> srun -n8 --pty --partition=angsd_class --mem=50G bash -i

> cd /athena/angsd/scratch/spv
> mkdir finalproject
> cd finalproject/
> wget <url> #repeated for all six samples
> ls #ensure all six samples were properly downloaded
  SRR8587778 SRR8587779 SRR8587780 SRR8587781 SRR8587782 SRR8587783
```

Observe the SRA files

As I downloaded each SRA file, I took advantage of the SRA website for each sample to look at the metadata and reads tabs. This provided me with helpful information that would be needed further downstream. My runs were of length 101, had 2 reads per spot, and were paired. Below is a screenshot of what this website looks like and where I can gather this information from. I looked at all of my samples to ensure they were the same across the board.



Converting SRA to FASTQ

Then, I loaded the SRA toolkit to convert all of my SRA files into Fastq files. To keep all of my files organized, I moved the SRA files into a separate directory, sra. As I was looking at the fastq-dump function, I began to

outline what options I would need to use. Assuming I would need both "–split-3" and "–split-spot", I double checked with Merv after finding some discrepencies online around "–split-spot." Merv advised that I just continue with "–split-files" for my only option, which provided me with 2 fastq files for each SRA.

```
mkdir sra
mv * ~/sra

spack load sra-toolkit
fastq-dump --split-spot --split-3 SRR8587781 #this is what I thought I had to do at first
fastq-dump --split-files SRR8587781 #repeated for all 6 files according to Merv
gzip *.fastq #compress files
```

Running FastQC & MultiQC

Once I obtained my FASTQ files, I ran FastQC, which is a very popular tool used to provide an overview of basic quality control metrics for raw next generation sequencing data. The output from FastQC, after analyzing a FASTQ file of sequence reads, is an html file that I pulled to my local desktop. I began by observing the per base sequence quality graph, which is a box-and-whisker plot showing aggregated quality score statistics at each position along all reads in the file. I know that the average quality score will steadily drop over the length of the read, but some of the scores were very low near the end, so I decided to trim my files using Trim_Galore, which is a script to automate quality and adapter trimming as well as quality control. Once I trimmed my FASTQ files, I ran FastQC once again.

```
spack load fastqc
spack load -r py-multiqc
spack load -r trimgalore

mkdir fastqc
fastqc *.fastq.gz --extract --outdir /athena/angsd/scratch/spv/finalproject/fastqc

mkdir fastq_trimmed/
trim_galore --illumina --paired SRR8587783_1.fastq.gz SRR8587783_2.fastq.gz --output_dir /athena/angsd/

mkdir fastqc_trimmed
fastqc *.fq.gz --extract --outdir /athena/angsd/scratch/spv/finalproject/fastqc_trimmed
multiqc /athena/angsd/scratch/spv/finalproject/fastqc_trimmed

#move all of the files to desktop for observance
scp [file] /home/spv4001
scp spv4001@aristotle.med.cornell.edu:/home/spv4001/* /Users/salonivishwakarma/desktop
```

QC after FastQC

1) How successful was the actual sequencing?
2) Did our library pep generate a faithful representation of the DNA/RNA molecules in our samples?

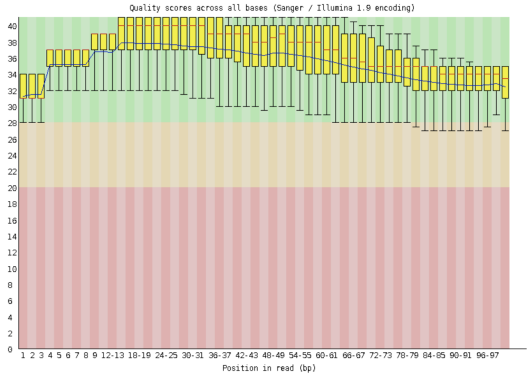
Overall, the FastQC was successful because all the FastQC files demonstrated consistently high base call confidence. I will answer these two questions by walking through one of my samples here (SRR8587778_2_val_2). It is important to recognize that all of the samples are more or less similar in results, but all have been trimmed to different amounts, so a little variation in the html files is expected. The html files for all FastQC and MultiQC will be located in my git repository, angsd-2020.

Basic Statistics

Measure	Value
Filename	SRR8587778_2_val_2.fq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	41683481
Sequences flagged as poor quality	0
Sequence length	20-101
%GC	47

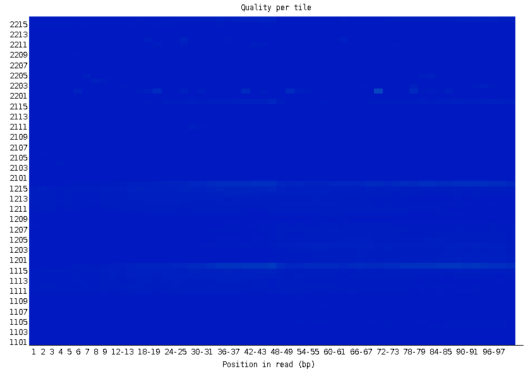
This will provide simple information about input FastQ file: name, type of quality score encoding, number of reads, read length, and GC content.

Per base sequence quality



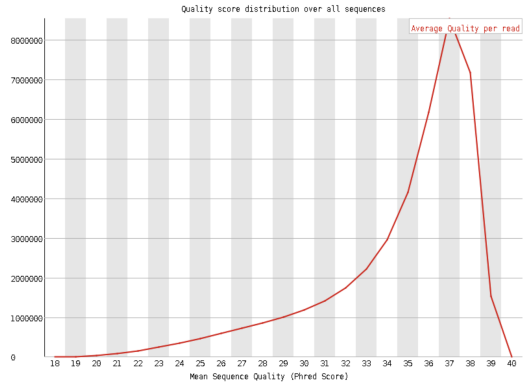
Per base sequence quality will show the aggregated quality score at each position along all the reads in the file. It is normal for files with Illumina sequencers for the median quality score to start out lower over the first few bases and then rise. The score will also tend to drop slightly over the length of the read. Lastly, one thing I noticed was that the average quality scores for read 1 were almost always slightly higher than for read 2.

Per tile sequence quality



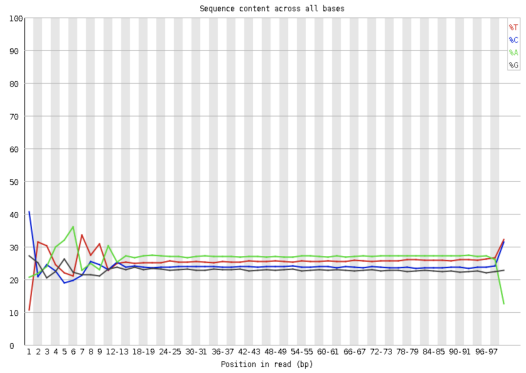
Per tile sequence quality reports the quality scores from each tile across all of your bases to see if there was a loss in quality associated with any certain part of the flowcell. Following a cold to hot scale where colder colors are positions where the quality was at or above average for that base in the run. Therefore, we should look for a plot that is blue all over or closest to it.

Per sequence quality scores



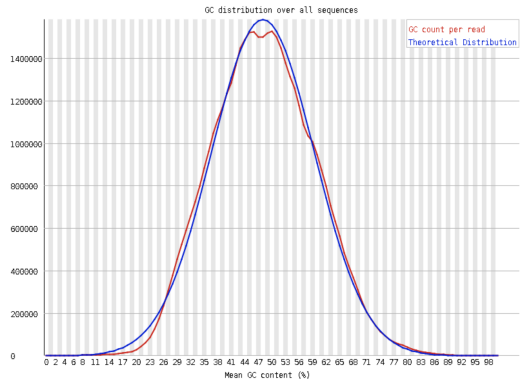
Per sequence quality scores is a plot of the total number of reads vs the average quality score over full length of that read. And this is good because the distribution of average read quality is fairly tight in the upper range of the plot.

Per base sequence content



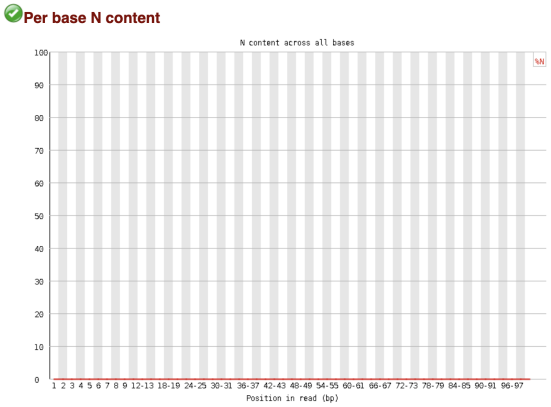
Per base sequence content is a plot that reports the percent of bases called for each of the four nucleotides at each position across all reads in the file. When I look at this plot, I look for relatively constant distribution of A, T, C, and G, at equal amounts. For most of the plot, this is the case. There is also an expected increase of variability in the end since the adapters were trimmed (possibility of one base getting cleaved). The variability in the early part of the plot could be potentially due to overrepresented sequences, biased fragmentation, or biased composition libraries.

Per sequence GC content

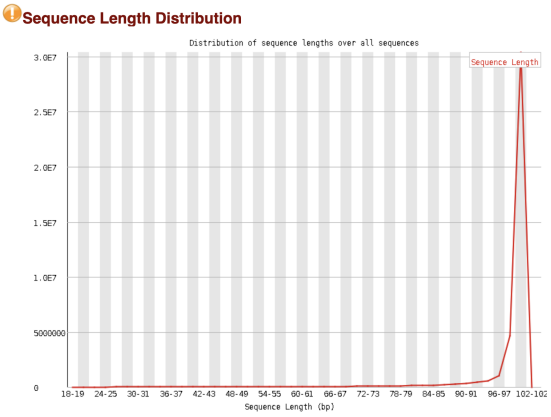


Per sequence GC content is a plot of the number of reads vs. GC% per read. There is an assumed uniform distribution (the blue plot). Our data should depict a GC content of all reads forming a uniform distribution

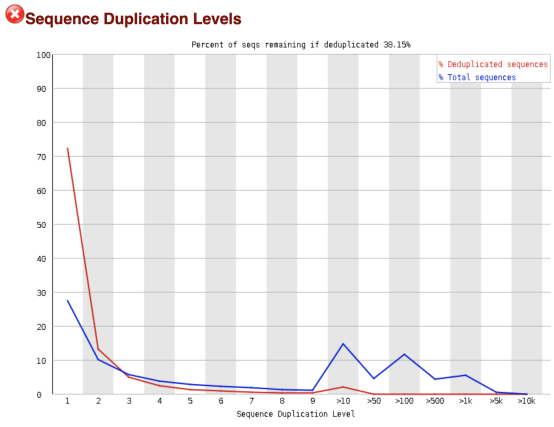
with the peak of the curve at the mean GC content.



Per Base N content plot demonstrates when the sequencer is unable to make a base call with sufficient confidence and thus, it will normally substitute an N rather than a conventional base call. So, I usually look for a low proportion of Ns.

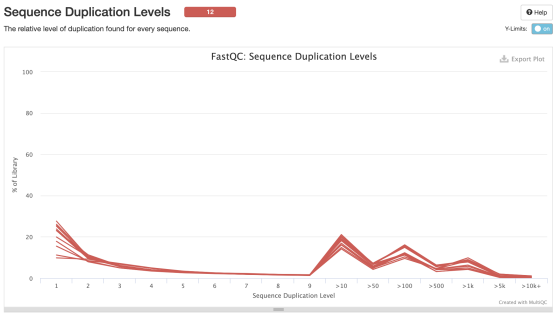


Sequence Length Distribution generates a graph showing the distribution of fragment sizes and often a single peak is okay.

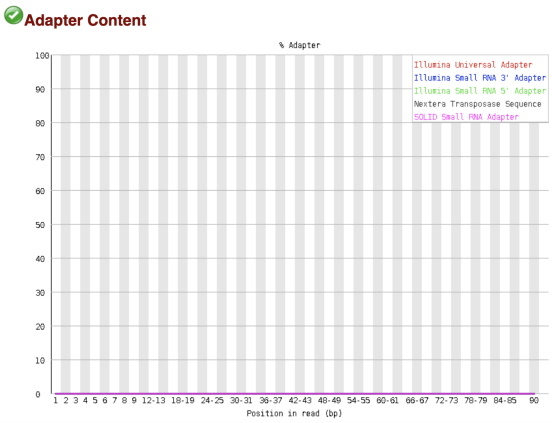


Sequence Duplication Levels will count the degree of duplication for every sequence in a library and creates a plot showing the relative number of sequences with different degrees of duplication. A low level of duplication might be because of a very high level of coverage of the target sequence. However, in this case, a high level of duplication can be due to either higher gene expression or indicating some kind of enrichment bias (For

example, that can be PCR over amplification). With RNA-Seq data, it is more likely that there are certain genes that are overly expressed and thus have more "duplicates".



By observing the multiqc for all of my fastqc_trimmed fles, I noticed that this was present in all 12 of my files (all six samples) and therefore, I continued my analysis almost confidently with the lack of variability between samples.



Adapter Content is a cumulative plot of the fraction of reads where the sequence library adapter sequence is identified at the indicated base position. This should all be 0 once the trimming has occured.

Alignment with STAR

STAR (Spliced Transcripts Alignment to a Reference) is an aligner designed to align RNA-seq data to a genome index using a strategy to account for spliced alignments. So, I can generate a human index genome by following the STAR guidelines, but the class ended up sharing one student's human genome index to save space on the cluster.

```
spack load star@2.7.0e
wget http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/hg38.fa.gz
gunzip hg38.fa.gz
wget http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/genes/hg38.ensGene.gtf.gz
gunzip hg38.ensGene.gtf.gz

STAR --runMode genomeGenerate \
      -- runThreadN 8 \
      -- genomeDir hg38_STARindex \
      -- genomeFastaFiles hg38.fa \
      -- sjdbGTFfile hg38.ensGene.gtf

#genome also accessible via Jess White's (classmate) directory to save space on cluster
```

```
/athena/angsd/scratch/jwh4001/project/hg38_STARindex
```

The following code is to run the actual alignment. Since my reads were paired, STAR took two files as inputs to indicate they were paired and outputted a single .bam file for that sample.

```
STAR --runMode alignReads \
     --runThreadN 8 \
     --genomeDir /athena/angsd/scratch/jwh4001/project/hg38_STARindex \
     --readFilesIn /athena/angsd/scratch/spv/finalproject/fastq_trimmed/SRR8587778_*.fq.gz \
     --readFilesCommand zcat \
     --outFileNamePrefix /athena/angsd/scratch/spv/finalproject/fastq_trimmed/star_test_trimmed/SRR8587
     --outSAMtype BAM SortedByCoordinate
```

QC of Aligned Reads

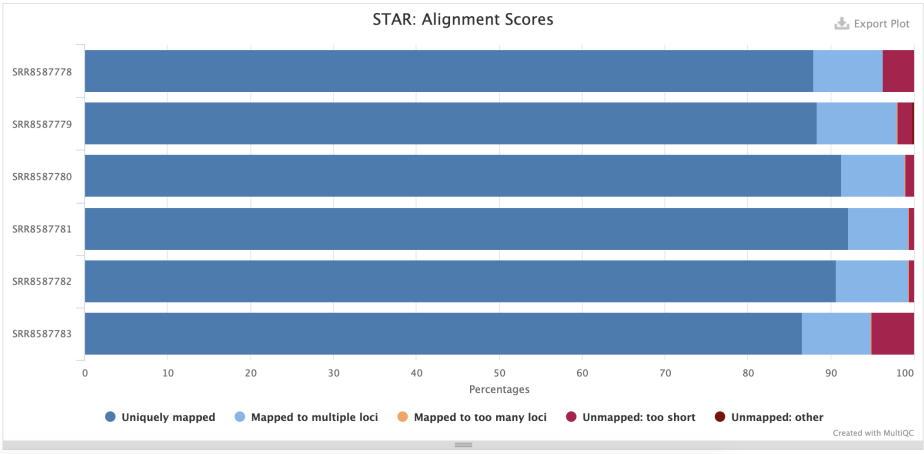
1) How many reads aligned? (By observing Log.final.out, STAR's log file)

```
head {SAMPLE}.Log.final.out
```

Log.final.out Output

Sample	Uniquely Mapped Reads %
SRR8587778 (BM-)	87.95
SRR8587779 (BM-)	88.39
SRR8587780 (BM-)	91.33
SRR8587781 (BM+)	92.10
SRR8587782 (BM+)	90.62
SRR8587783 (BM+)	86.57

The data mostly mapped to unique reads in the genome. To also look at the uniquely mapped vs the unmapped vs the multi mapped, which should ideally add up to 100%, I ran mutliQC to gain more information about my bam files to obtain that information:



2) How well did the reads align? (By observing samtools flagstat file)

```
spack load samtools@1.9%gcc@6.3.0
samtools index *.bam
samtools stats SRR8587781.Aligned.sortedByCoord.out.bam > SRR8587781.stats #get stats
samtools stats SRR8587781.Aligned.sortedByCoord.out.bam > SRR8587781.stats #get flagstats

grep ^SN SRR8587781.stats | cut -f 2-
```

```
cat *.flagstats
```

Flagstats Output

Sample	Mapped Reads %	Properly Paired %
SRR8587778 (BM-)	100.00	99.92
SRR8587779 (BM-)	100.00	99.70
SRR8587780 (BM-)	100.00	99.86
SRR8587781 (BM+)	100.00	99.92
SRR8587782 (BM+)	100.00	99.90
SRR8587783 (BM+)	100.00	99.88

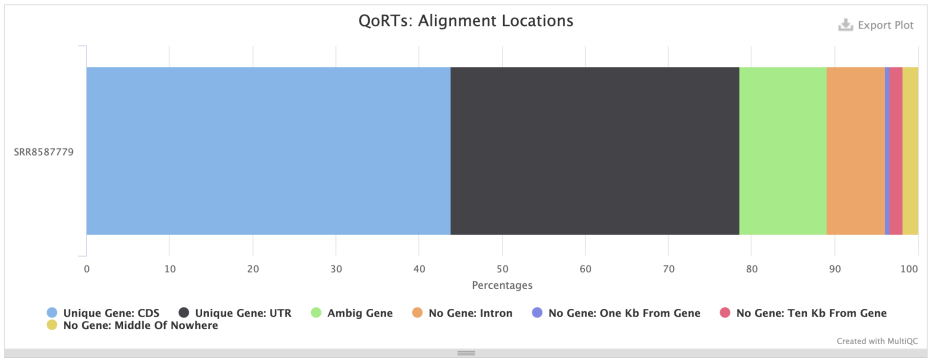
The 100%, at first suspiciously too perfect, is expected since flagstats takes in my bam files only. And if my bam files were produced using STAR that only stores the mapped reads, then inherently, they will all 100% be mapped.

3) Did we capture mostly exonic RNA? (RSeQC's read_distribution.py, QoRTS)

QoRTs is an alternative to RSeQC and offers gene diversity plot and more fine-grained plots where genes are stratified by expression strength. For paired-end data reads must be sorted. By default, QoRTs assumes the input bam file consists of paired-end data. For single-end data, the –singleEnded option must be used.

```
spack find | grep -i qorts
spack load qorts@1.2.42
$ QORTS_LOC=`spack location -i qorts`

for SAMPLE in SRR8587778 SRR8587779 SRR8587780 SRR8587781 SRR8587782 SRR8587783
do
  java -Xmx5G -jar ${QORTS_LOC}/bin/QoRTs.jar QC \
  --generatePdfReport \
  /athena/angsd/scratch/spv/finalproject/fastq_trimmed/star_test_trimmed/${SAMPLE}*Aligned.sortedByCoor
  /athena/angsd/scratch/spv/finalproject/fastq_trimmed/star_test_trimmed/hg38.ensGene.gtf \
  /athena/angsd/scratch/spv/finalproject/fastq_trimmed/star_test_trimmed/qorts_final/${SAMPLE}/
done
```

This file from multiqc also shows how the majority of the read is from exonic data.



featureCounts

After sequencing, I decided to count read-gene overlaps with featureCounts post-alignment. Since my reads are paired, each pair of reads defines a DNA or RNA fragment bookended by the two reads. In this case, featureCounts can be instructed to count fragments rather than reads. featureCounts automatically sorts

reads by name if paired reads are not in consecutive positions in the SAM files. My options are explained below:

- -p : is pair-ended data
- -t : Only rows which have the matched feature type in the provided GTF annotation file will be included for read counting. 'exon' by default
- -g : 'gene id' by default.
- -a : annotation file
- -o : output file

```
spack load subread
featureCounts -p -t exon -g gene_id -a hg38.ensGene.gtf -o featurecounts_genes.txt *.bam

#this file is too massive to bring into Rmarkdown
wc -l final_project_genes.txt
  64254 final_project_genes.txt
ls -lh final_project_genes.txt
  -rw-r--r-- 1 spv4001 angsd_class 34M Apr  7 01:48 final_project_genes.txt
```

Need to Trim featureCounts Output

Since the file was too big initially for RMarkdown to take in (34M), I sought help from my professors to trim my file, which was achieved by removing all columns except GeneID and my samples. Once trimmed, my file was only 2M and could easily be loaded into RMarkdown.

```
#cut this file to remove superfluous columns
cat final_project_genes.txt | cut -f 1,7-12 > featureCounts_finalproj.txt

wc -l featureCounts_finalproj.txt
  64254 featureCounts_finalproj.txt
ls -lh featureCounts_finalproj.txt
  -rw-r--r-- 1 spv4001 angsd_class 2.0M Apr  9 14:56 featureCounts_finalproj.txt

scp featureCounts_finalproj.txt /home/spv4001
scp spv4001@aristotle.med.cornell.edu:/home/spv4001/featureCounts_finalproj.txt /Users/salonivishwakarma
```

Load featureCounts file into R

With my data online, HTSeq data was also provided, which is an alternative to featureCounts. I loaded it just in case to see whether the researchers and I did the alignment in the same way. If so, then we would have the same numbers.

```
library(ggplot2) # for making plots
library(magrittr)

#read txt file as table
readcounts <- read.table("featureCounts_finalproj.txt", header=TRUE)
HTSeq_data = read.table("GSE126548_HTSeq_count.txt", header=TRUE) #Additional file from HTSeq from proj

#str(readcounts)
```

Change the Column Names in readcounts_sym

```
orig_names <- names(readcounts)
names(readcounts) <- c("Geneid", "78_BM+", "79_BM+", "80_BM+", "81_BM-", "82_BM-", "83_BM-")

#change from factors to characters
readcounts$Geneid <- as.character(readcounts$Geneid)
```

Change from Ensembl IDs to Gene Symbols

There are many different databases I can use and different packages I have tried and show further below. My original feature counts file gives me 64252 rows (64252 ensembl IDs) and with each data base that I am using to select gene symbols from, I end up getting either fewer rows or more rows with NA values, indicating that those (~1000-4000) Ensembl IDs do not have Gene Symbols.

Therefore, I restricted the analysis to genes with unique annotated gene symbols. I continue my analysis working with the data set that provided me with the largest possible subset (EnsDb.Hsapiens.v79).

```
#if (!requireNamespace("BiocManager", quietly = TRUE))
#    install.packages("BiocManager")
#BiocManager::install("EnsDb.Hsapiens.v79")
library("EnsDb.Hsapiens.v79")

#store all of my Ensemble IDs
ensemblIDS <- c(readcounts$Geneid)

#load the Gene Symbols for those Ensemble IDS
symbols <- select(EnsDb.Hsapiens.v79, key=ensemblIDS, columns=c("SYMBOL"), keytype="GENEID")
colnames(symbols) <- c("Geneid", "Symbol")

#merge to include readcounts with symbols
readcounts_sym <- merge(readcounts, symbols, by="Geneid")
```

However, I still want to know what was left out and so it is important that I characterize that in broad strokes. So, if I put my readcounts in matrix form, I can more easily count what would be left out.

```
## index of genes missing IDs
idx.missing <- which(!(ensemblIDS %in% symbols$Geneid))

## convert to matrix with rownames
mat.counts <- readcounts[,-1] %>% as.matrix %>% set_rownames(readcounts[,1])

## percentages of total counts in excluded genes
colSums(mat.counts[idx.missing,])/colSums(mat.counts)
```

```
##       78_BM+       79_BM+       80_BM+       81_BM-       82_BM-       83_BM-
## 0.0005919410 0.0005555599 0.0005556121 0.0006161443 0.0003465177 0.0006569590
```

```
#Alternative routes explored:

    BiocManager::install("org.Hs.eg.db")
    BiocManager::install("AnnotationDbi")
    BiocManager::install("biomaRt")
    BiocManager::install("EnsDb.Hsapiens.v86")
    library("org.Hs.eg.db")
    library("AnnotationDbi")
    library("biomaRt")
    library("EnsDb.Hsapiens.v86")

    #org.Hs.ed.qb, 64644 row
    symbols_2 <- select(org.Hs.eg.db, key=ensemblIDS, columns=c("SYMBOL"), keytype="ENSEMBL")

    #bioMART, 60011 rows
    mart<- useDataset("hsapiens_gene_ensembl", useMart("ENSEMBL_MART_ENSEMBL"))
    symbols_3 <- getBM(filters="ensembl_gene_id", attributes=c("ensembl_gene_id", "hgnc_symbol"),
```

```
values=ensemblIDS, mart=mart)

#EnsDb.Hsapiens.v86, 60704 rows
symbols_4 <- select(EnsDb.Hsapiens.v86, key=ensemblIDS, columns=c("SYMBOL"), keytype="GENEID")
```

DESeq Analysis

Now, I will use the DESeq2 package to normalize the samples for differences in their sequencing depth and to explore them. Therefore, I generate a DESeqDataSet, an R object class, which combines data.frames and one or more matrices into one object. The data.frames typically contain metadata about the samples and genes (e.g. gene IDs, sample conditions), while the matrices contain the expression values.

```
#BiocManager::install("DESeq2")
library(DESeq2)
library(scater)
```

I quickly learned that I had gene duplicates which means that the different Ensemble IDs mapped to the same Gene Symbol. In addition to losing some Ensemble IDs (~1000), I had many duplicates in my genes.

```
length(readcounts_sym$Symbol)
  64252
length(unique(readcounts_sym$Symbol))
  58329
```

Instead of getting rid of 8,000 more Ensembl IDs, I decided to add a "dup" and a random number in a range. This might be misleading since I won't know just by looking at the number following the gene name. But, I will be able to group these at another point. Then, I took out the symbols from readcounts_sym and stored them as a character, which preserved the order in the list so I know which Ensembl ID it matches with. Then I used the uniquifyFeatureNames to add a dup followed by a random number to indicate a unique gene.

```
#identifying symbols
just_symbols <- c(readcounts_sym$Symbol)

#making sure each Ensembl id that has a gene ID is represented
symbols_unique <- uniquifyFeatureNames(ID=paste0("dup", 1:63748), names = just_symbols)

#gene IDs should be stored as row.names
row.names(readcounts_sym) <- symbols_unique

#double check that I have formed the counts correctly
colnames(readcounts_sym)
```

```
## [1] "Geneid" "78_BM+" "79_BM+" "80_BM+" "81_BM-" "82_BM-" "83_BM-" "Symbol"
```

```
#store geneIDS that you are working with
counts_ensembleIDS <- c(readcounts_sym$Geneid)

#remove geneid and symbol column from counts
readcounts_sym  <- readcounts_sym[-c(1, 8)]

#final check that readcounts looks good
head(readcounts_sym)
```

```
##          78_BM+ 79_BM+ 80_BM+ 81_BM- 82_BM- 83_BM-
## TSPAN6     4373   8160   5522   6557   9814   6536
```

```
## TNMD          0      0      0      0      2     70
## DPM1       4615   2711   6110   3395   6432   2278
## SCYL3      1466    391    662   1211    303    659
## C1orf112    617     99    761    598    459    138
## FGR        3132    220    305   2264    412   1727
```

Therefore, I make two tables: countData and colData. colData should be a data.frame, where the rows directly match the columns of the count data. And countData is my readcounts_sym data.frame. By making a DESeq object, I can then explore a lot of questions.

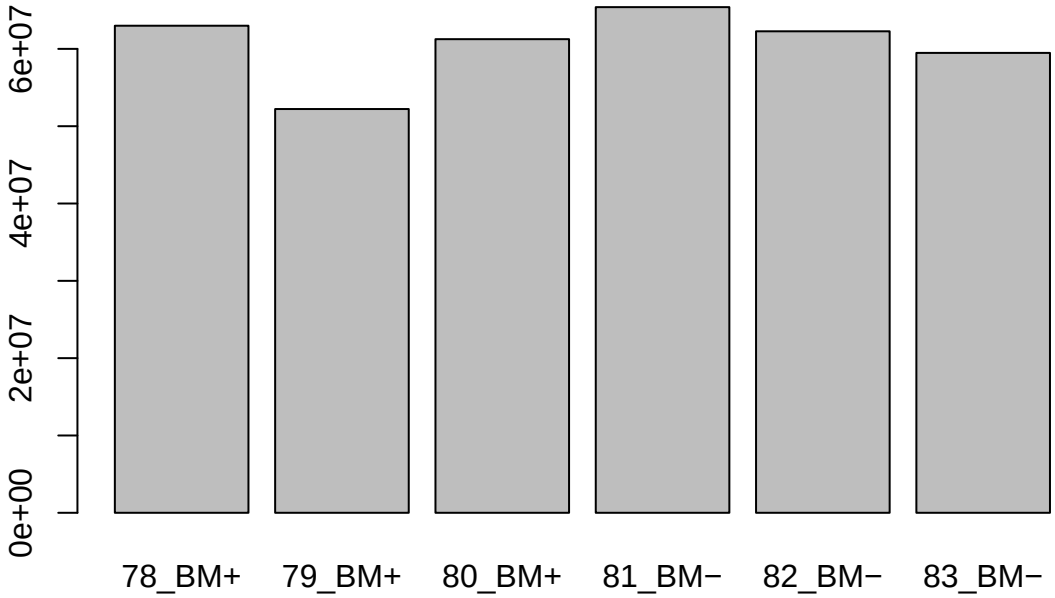
```
sample_info <- DataFrame(condition = names(readcounts_sym),row.names = names(readcounts_sym))
DESeq.ds <- DESeqDataSetFromMatrix(countData = readcounts_sym, colData = sample_info, design = ~ conditi
```

Question: How many reads were sequenced for each sample (aka the library sizes)?

```
colSums(counts(DESeq.ds))
```

```
##   78_BM+   79_BM+   80_BM+   81_BM-   82_BM-   83_BM-
## 62992617 52217366 61255106 65397276 62272482 59479071
```

```
colSums(counts(DESeq.ds)) %>% barplot
```



This barplot shows that the sequencing depth of all of the samples are different, especially for sample "79_BM+", which is why normalizing for sequencing depth becomes a very important step further downstream.

Remove genes with no reads

```
keep_genes <- rowSums(counts(DESeq.ds)) > 0
DESeq.ds <- DESeq.ds[ keep_genes, ]
dim(DESeq.ds)
```

```
## [1] 40876     6
```

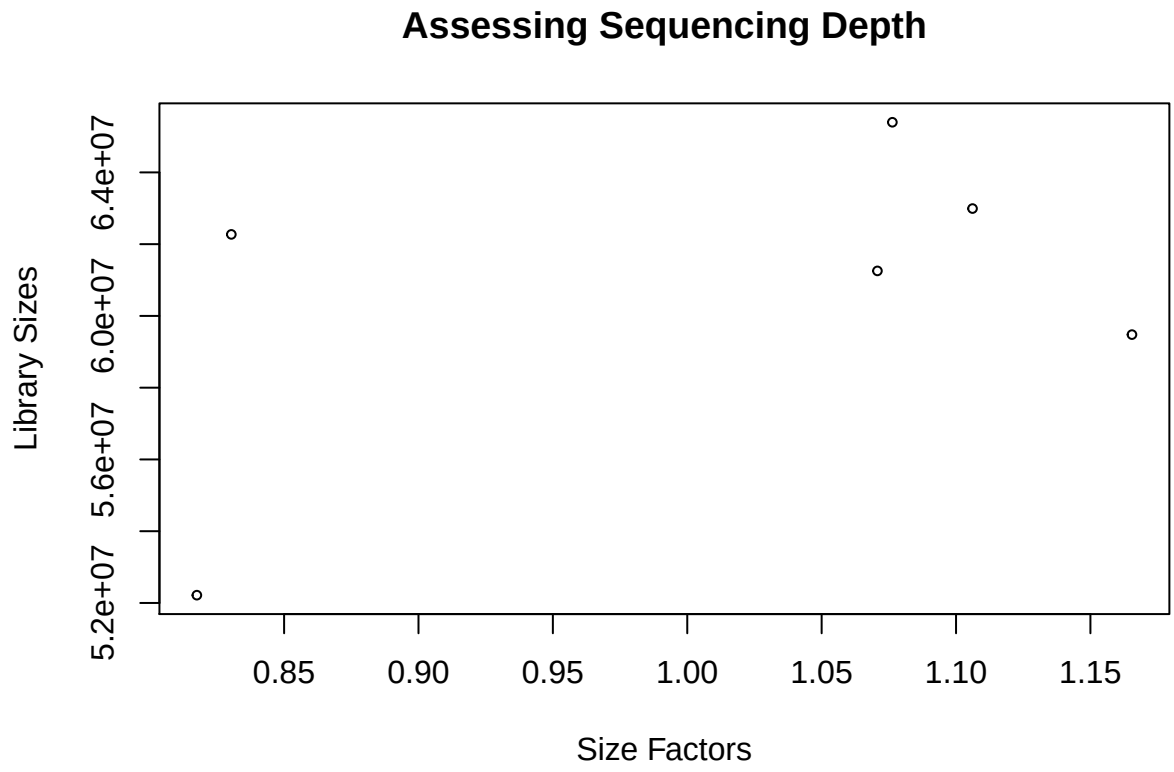
Before filtering, dim(DESeq.ds) showed that our object had the dimensions of 63748 by 6. Now, the dimensions are 40876 by 6, which means that 22,872 genes had no counts in any of my samples. Now, I have a more condensed data set to work with.

Assessing Sequencing Depth

I use the function, estimateSizeFactors(), to observe sequencing depth normalization because some samples

may have been sequenced to a different depth, so when comparing gene expression, it is important to normalize that data. As you can see below, my six samples have different sequencing depth.

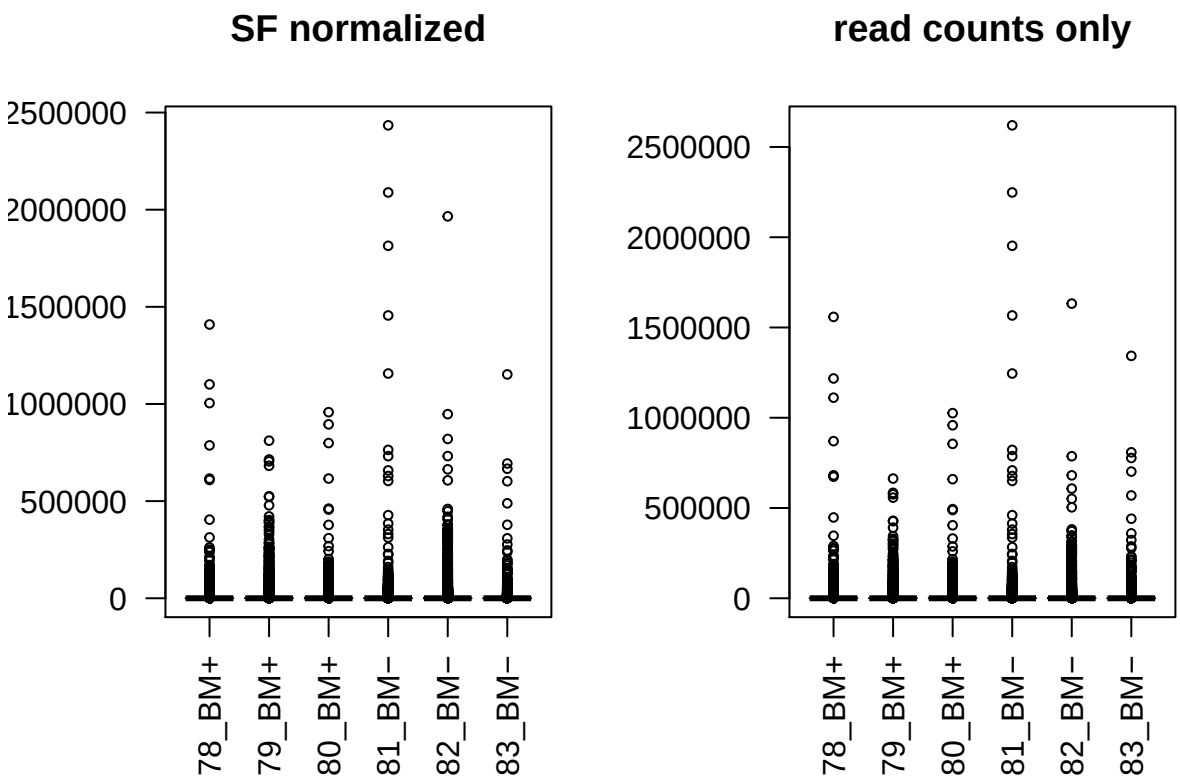
```
DESeq.ds <- estimateSizeFactors(DESeq.ds) # calculate SFs, add them to object

plot( sizeFactors(DESeq.ds), colSums(counts(DESeq.ds)), # assess them
ylab = "Library Sizes", xlab = "Size Factors", cex = .6, main = "Assessing Sequencing Depth")
```



Normalizing for Sequencing Depth and RNA Composition Differences

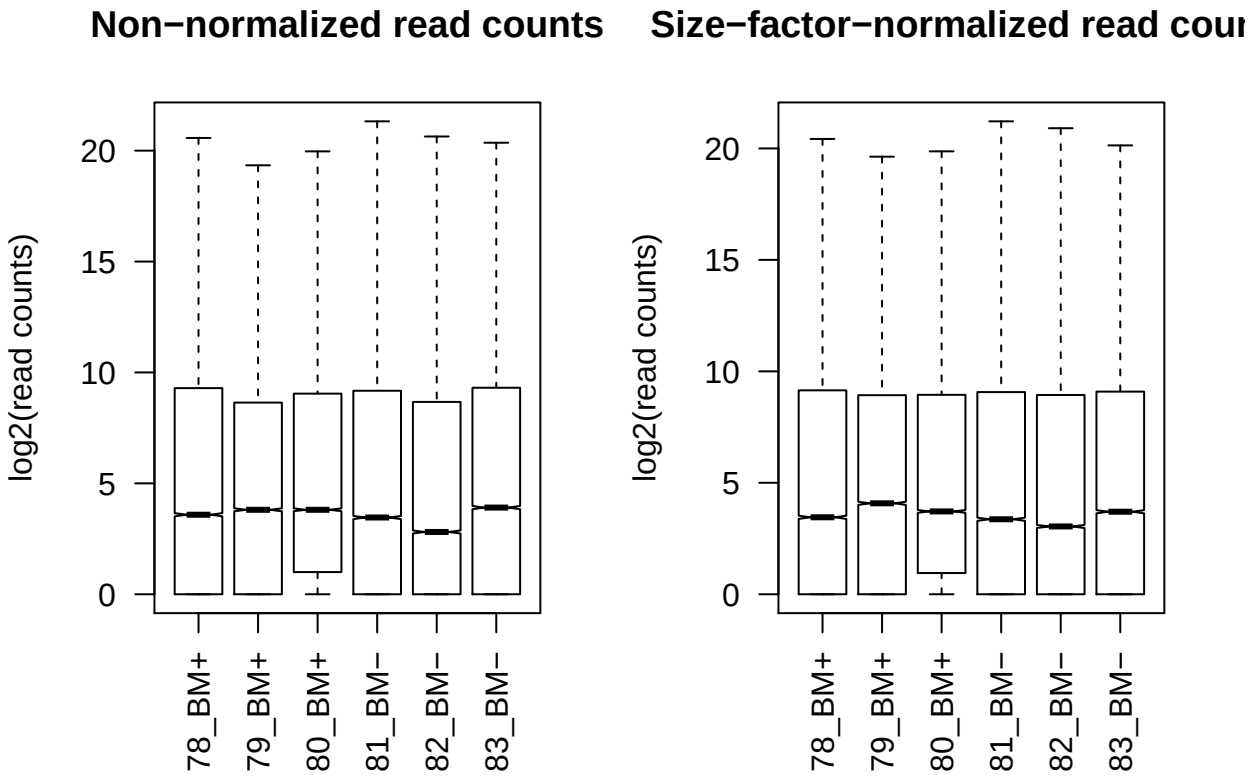
The read counts normalized for sequencing depth can be accessed via counts(..., normalized = TRUE). So, I plotted blox plots showing whether the normalization helped adjust global differences between the six samples.

```
par(mfrow=c(1,2))

#extracting normalized counts
counts.sf_normalized <- counts(DESeq.ds, normalized=TRUE)

## adding the boxplots
boxplot(counts.sf_normalized, main = "SF normalized", cex = .6, las = 2)
boxplot(counts(DESeq.ds), main = "read counts only", cex = .6, las =2)
```



However, with the range of read counts on the y axis, I can't really see anything. Therefore, I transformed the normalized read counts to bring them onto more similar scales. With these plots, I can see that the data on the right has been normalized.

```
par(mfrow=c(1,2)) #to plot two box plots

#non-normalized
boxplot(log2(counts(DESeq.ds)+1), notch=TRUE, main = "Non-normalized read counts", ylab="log2(read coun

#size-factor normalized values
boxplot(log2(counts(DESeq.ds, normalize= TRUE) +1), notch=TRUE, main = "Size-factor-normalized read cou
```



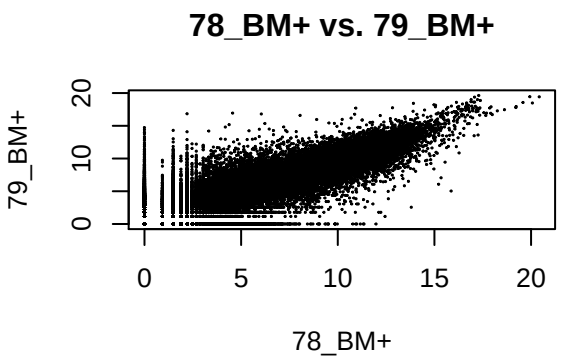
Scatterplot of Log Normalized Counts against each other

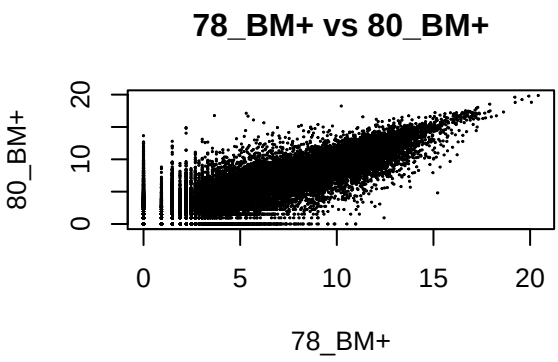
I plot a scatterplot to observe how well the actual values correlate which each other per sample and gene. I looked at the BM+ samples amongst one another and then the BM- samples amongst one another.

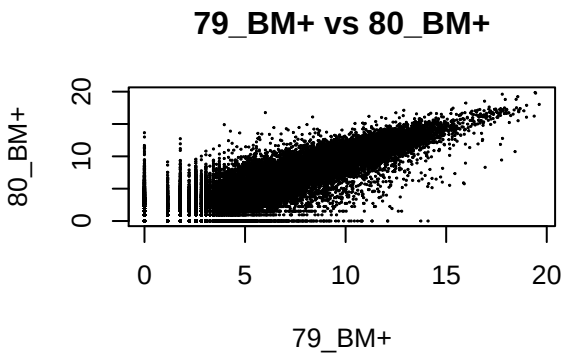
```
log.counts <- log2(counts(DESeq.ds, normalized = FALSE) + 1) #assign the values to a distinct matrix
assay(DESeq.ds, "log.counts") <- log2(counts(DESeq.ds, normalized = FALSE) + 1)

# normalized read counts
log.norm.counts <- log2(counts(DESeq.ds, normalized=TRUE) + 1)
assay(DESeq.ds, "log.norm.counts") <- log.norm.counts
```

Plotting BM+ samples against one another:

```
par(mfrow=c(2,2))
DESeq.ds[, c("78_BM+","79_BM+")] %>% assay(., "log.norm.counts") %>% plot(., cex=.1, main = "78_BM+ vs.
DESeq.ds[, c("78_BM+","80_BM+")] %>% assay(., "log.norm.counts") %>% plot(., cex=.1, main = "78_BM+ vs 8
DESeq.ds[, c("79_BM+","80_BM+")] %>% assay(., "log.norm.counts") %>% plot(., cex=.1, main = "79_BM+ vs 8
```

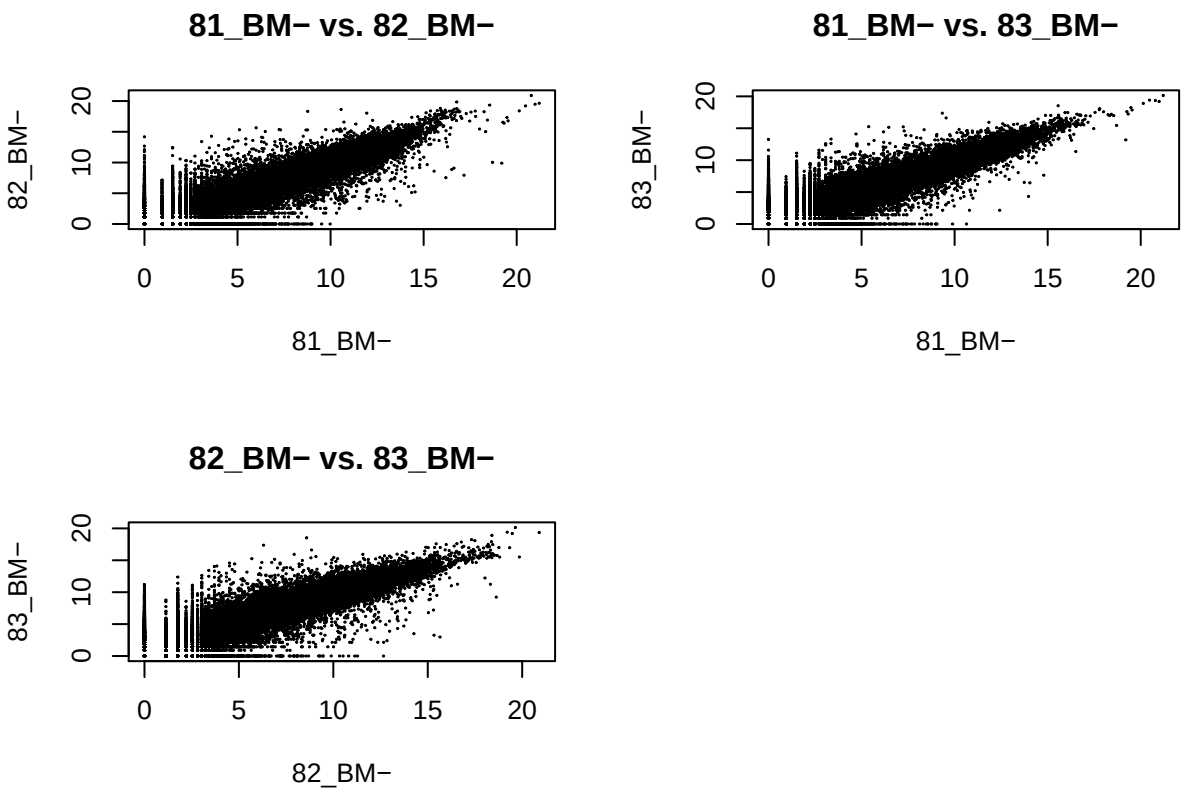






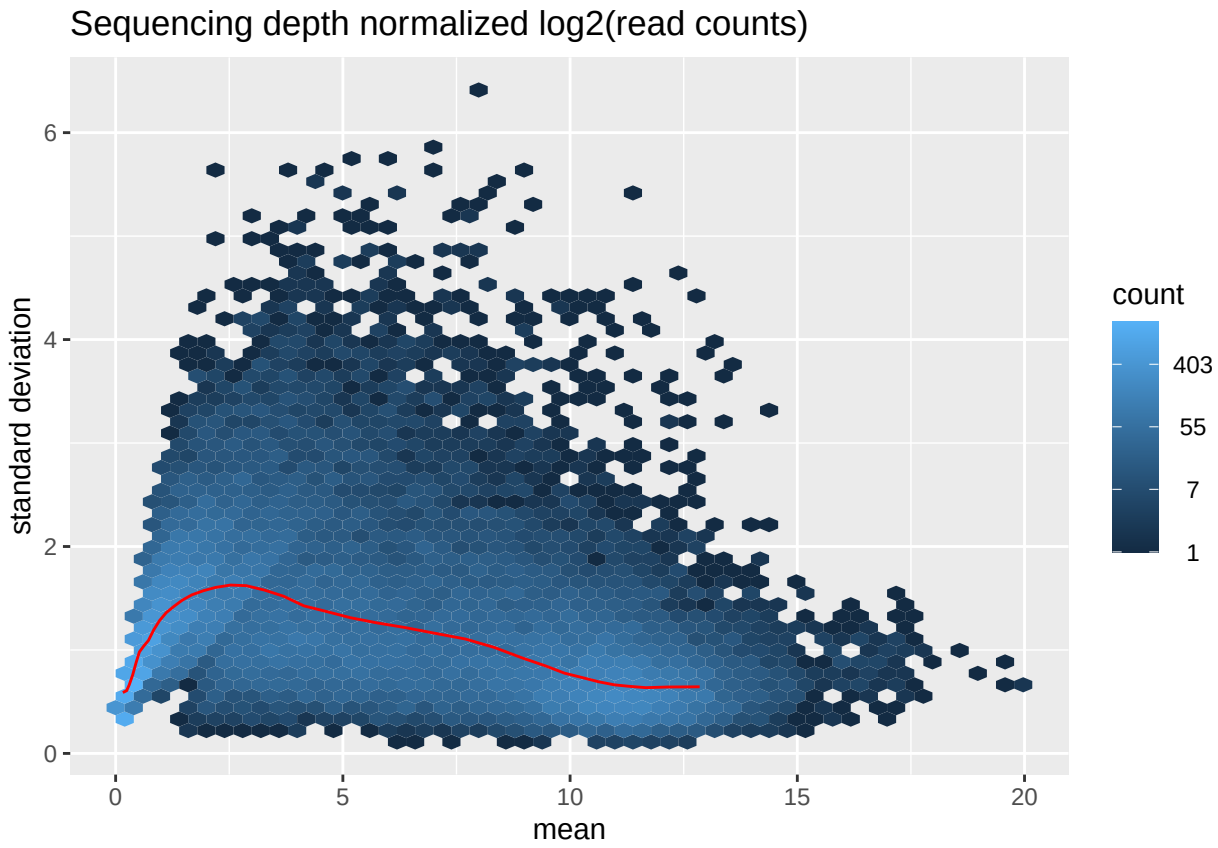
Plotting BM- samples against one another:

```
par(mfrow=c(2,2))
DESeq.ds[, c("81_BM-","82_BM-")] %>% assay(., "log.norm.counts") %>% plot(., cex=.1, main = "81_BM- vs.
DESeq.ds[, c("81_BM-","83_BM-")] %>% assay(., "log.norm.counts") %>% plot(., cex=.1, main = "81_BM- vs.
DESeq.ds[, c("82_BM-","83_BM-")] %>% assay(., "log.norm.counts") %>% plot(., cex=.1, main = "82_BM- vs.
```



In these plots, every dot is one gene. The fanning out of the points in the lower left corner indicates that read counts correlate less well between replicates when they are low. This observation indicates that the standard deviation of the expression levels may depend on the mean: the lower the mean read counts per gene, the higher the standard deviation. Between the BM+ and the BM- samples, it seems that there is less correlation between low replicates and read counts in the BM+ samples since the dots are more spread out.

```
#generate the base meanSdPlot using sequencing depth normalized log2(read counts)
log.norm.counts <- log2(counts(DESeq.ds, normalized=TRUE) + 1)

#set up ploting frames
par(mfrow=c(1,1))

#generate the plot
msd_plot <- vsn::meanSdPlot(log.norm.counts, ranks=FALSE, plot = FALSE)
msd_plot$gg + ggtitle("Sequencing depth normalized log2(read counts)") + ylab("standard deviation")
```

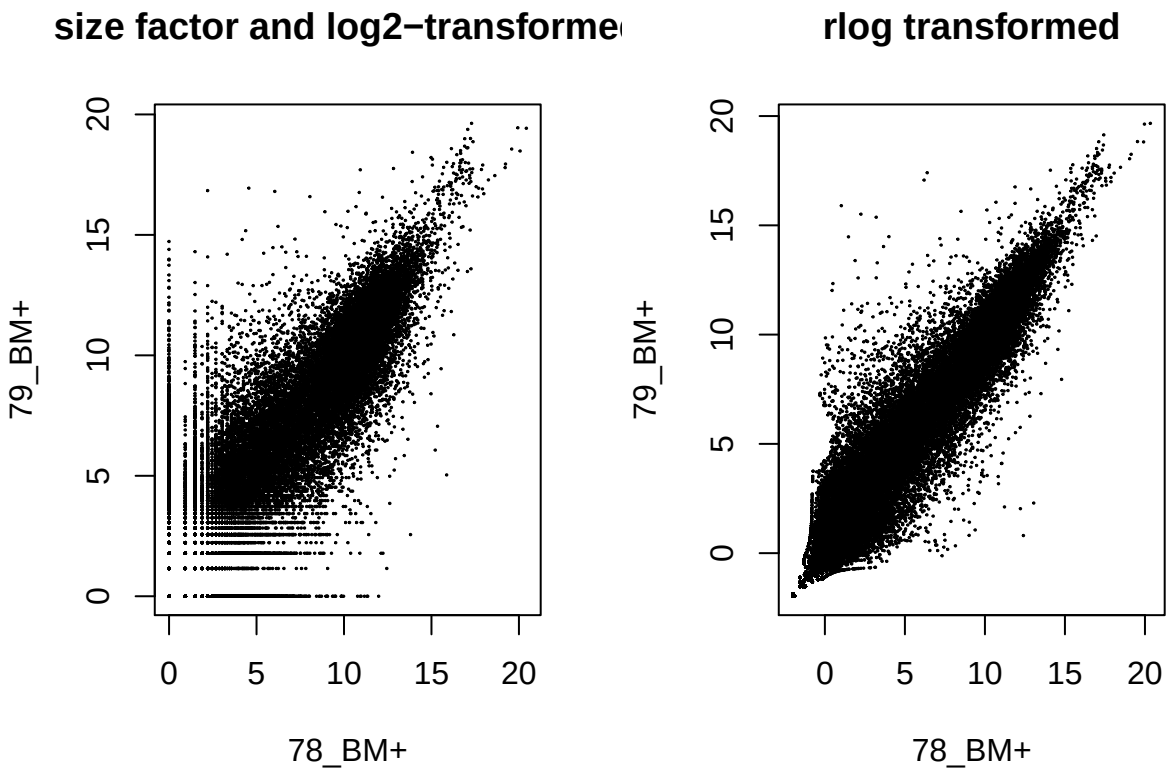


This is a meanSdPlot, which can help answer the question of whether or not the standard deviation of the expression levels may depend on the mean. The red dots depict the running median estimator. If there is no variance-mean dependence, then the line formed by the red dots should be approximately horizontal. That is not the case here: this plot shows that there is a lot of variance-mean dependence for genes with low read counts. This means that the data is heteroskedastic.

Rlog Function

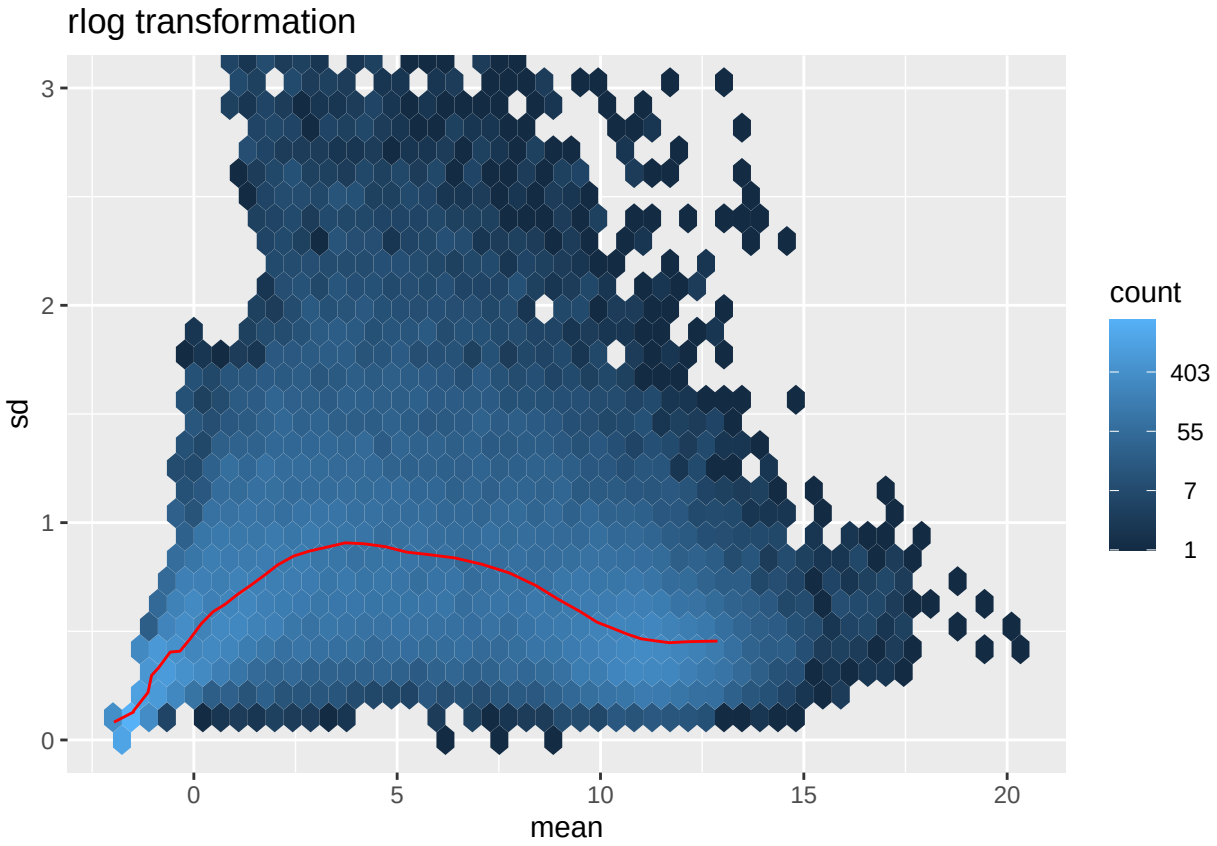
rlog is a function that is considered an optimized method for RNA-seq read counts to reduce the dependence of the variance on the mean. The function transforms the read counts to the log2 scale and minimizes the difference between samples for rows with small counts. This is important because it considers the difference between library sizes of the samples. Here, I use rlog to reduce the dependence of the variance on the mean.

```
## this actually generates a different type of object!
DESeq.rlog <- rlog(DESeq.ds, blind = TRUE)

par(mfrow=c(1,2))
plot(log.norm.counts[,1:2], cex=.1,
main = "size factor and log2-transformed")

## the rlog-transformed counts are stored in the accessor "assay"
plot(assay(DESeq.rlog)[,1], assay(DESeq.rlog)[,2], cex=.1, main = "rlog transformed", xlab = colnames(as
```



Here, I can see that the rlog transformation was successful. The variance that is higher for small read counts has tightened significantly from the left to the right plot.

```
rlog.norm.counts <- assay(DESeq.rlog)
## rlog-transformed read counts
msd_plot <- vsn::meanSdPlot( rlog.norm.counts, ranks=FALSE, plot = FALSE)
msd_plot$gg + ggtitle("rlog transformation") + coord_cartesian(ylim = c(0,3))
```



Although it's not perfectly horizontal, this result has improved tremendously. Following Professor Dündar's workflow given in class, I was able to adjust my data for differences in sequencing depth, differences in RNA composition, heteroskedasticity, and large dynamic range. Now, I saved that as an RData, so I can proceed to analyze it further.

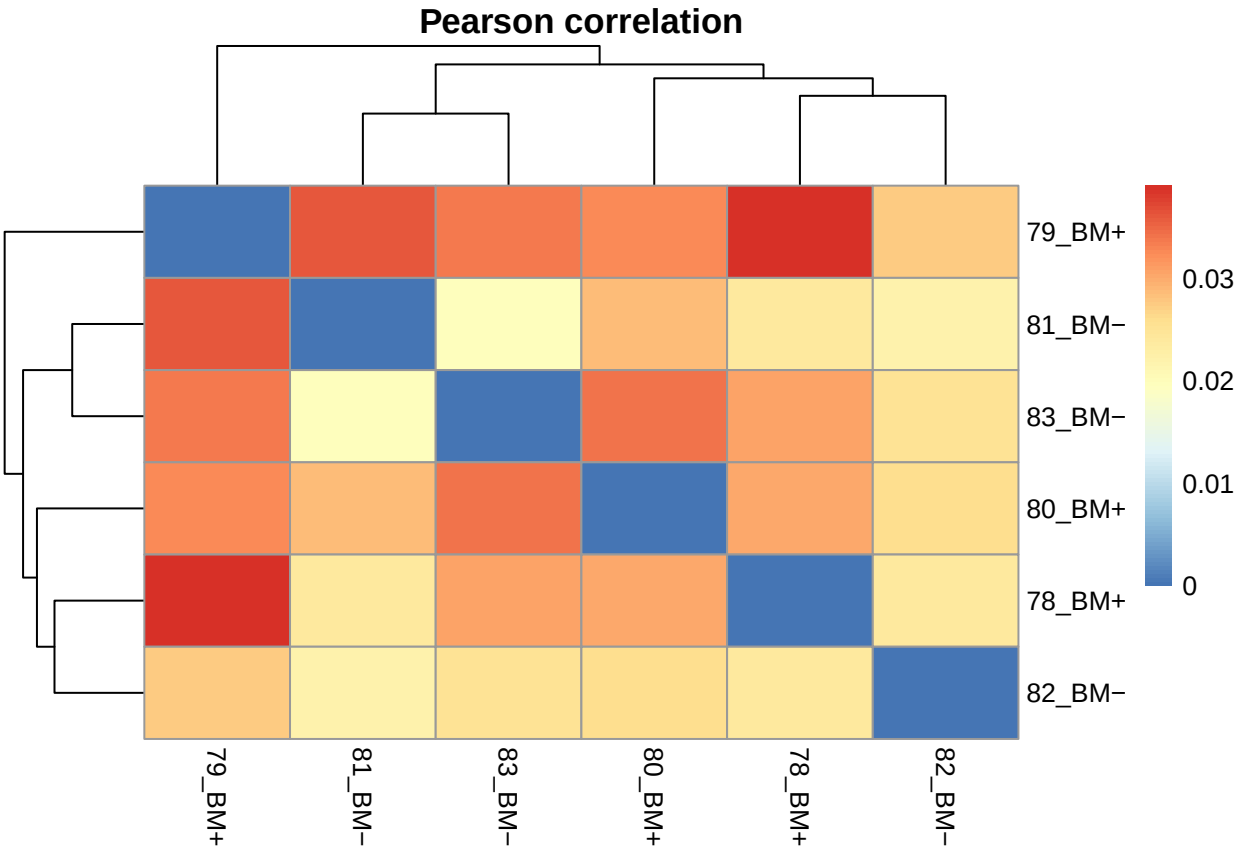
```
#save file
save.image(file = "SV_FinalProject.RData")
```

Exploratory Plots

Sample Clustering with Pearson correlation

Once I obtained the normalized read counts, I wanted to see if the BM+ samples are more similar to each other than the BM- samples. I used the pearson correlation coefficient because it is a measure of the strength of the linear relationship between two variables and is often used to assess the similarity of RNA-seq samples in a pair-wise fashion. Mathematically, this is the covariance of two variables divided by the product of their standard deviation. When reading the heatmap below, values closer to zero means there is no linear trend between the two variables. As the correlation is closer to 1, it suggests that those two samples are more positively correlated. Below, you can see that 79_BM+ and 78_BM+ are two samples that are the most correlated with one another. And that is expected because they are two brain metasasis lung cancer samples, but the other samples are not correlated in any pattern.

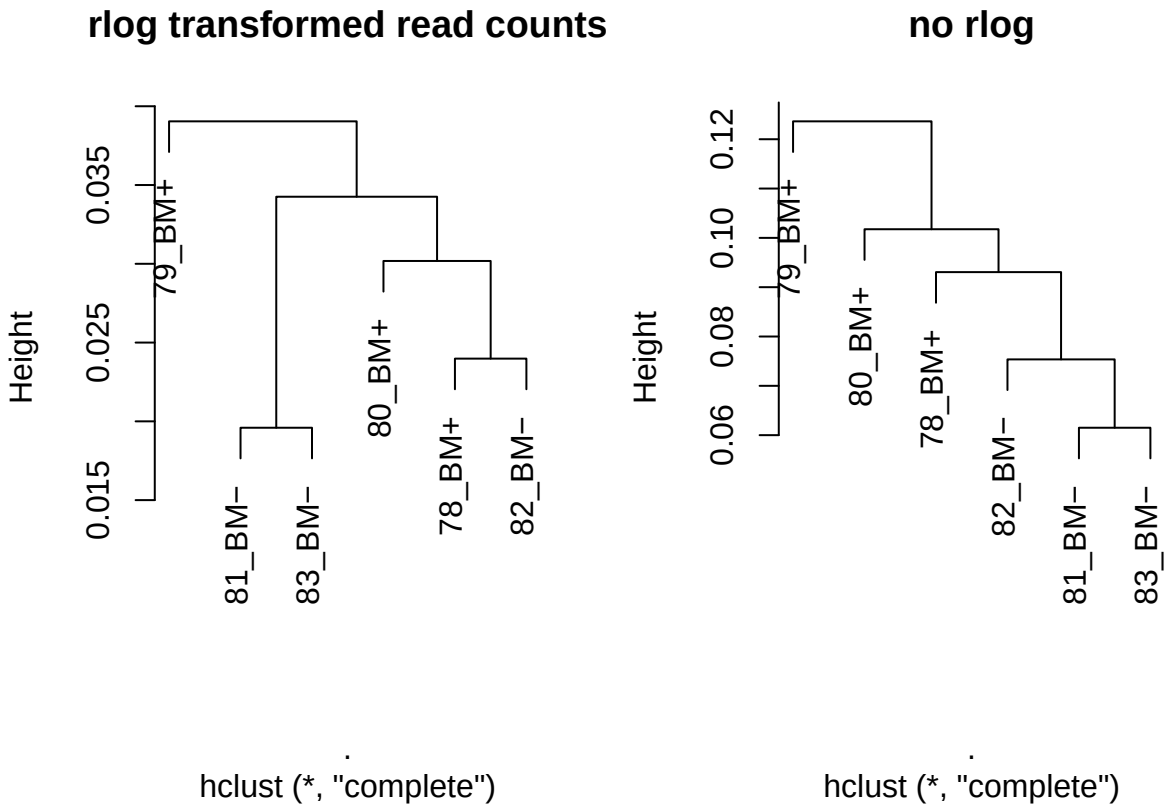
```
load("SV_FinalProject.RData")
corr_coeff <- cor(rlog.norm.counts, method = "pearson")
as.dist(1-corr_coeff, upper = TRUE) %>% as.matrix %>% pheatmap::pheatmap(., main = "Pearson correlation
```



Dendogram

A dendogram is a diagram that shows the hierarchical relationship between objects and works out the best way to allocate objects to clusters. I plotted the dendrogram above to compare the effects of the rlog transformation. It is interesting how much the normalization was able to flip over the samples 81_BM- and 83_BM-, but this might be since 81_BM- had the greatest sequencing depth, so the normalizatino would affect it the most.

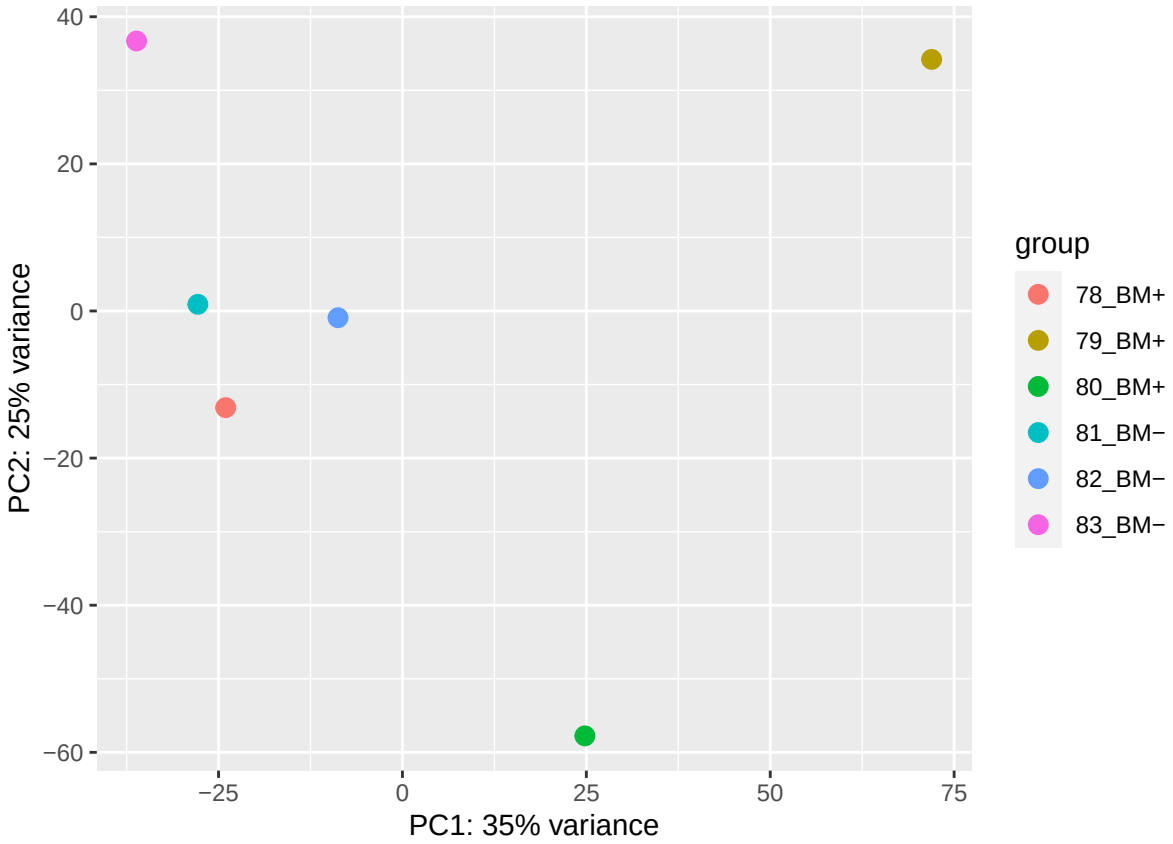
```
par(mfrow=c(1,2))

# Pearson corr. for rlog.norm values
as.dist(1 - corr_coeff) %>% hclust %>% plot( ., labels = colnames(rlog.norm.counts), main = "rlog transf

# Pearson corr. for log.norm.values
as.dist( 1 - cor(log.norm.counts, method = "pearson")) %>% hclust %>% plot( ., labels = colnames(log.no
```



PCA

Principal Component Analysis is a tool used to visualize genetic distance and relatedness between populations or samples. By projecting my data into a smaller space, PCA reduces the dimensionality of the feature space. So I use the prcpomp function to run PCA on my rlog samples data.

```
rv <- rowVars(assay(DESeq.rlog)) # equivalent to rowVars(rlog.norm.counts)
top_variable <- order(rv, decreasing = TRUE)[seq_len(500)]
pca <- prcomp(t(assay(DESeq.rlog)[top_variable, ]))
plotPCA(DESeq.rlog)
```



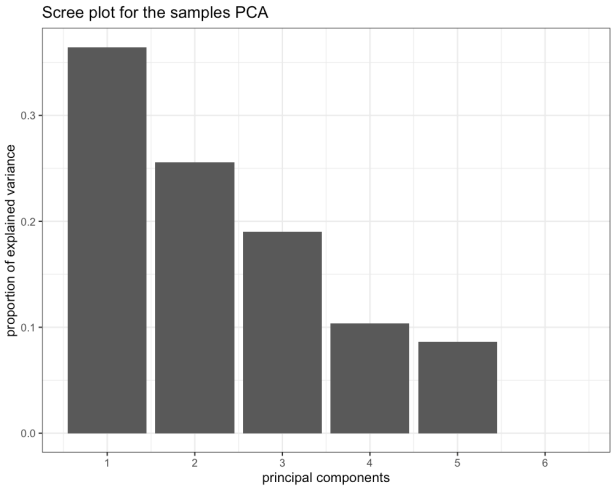
In PCA, PC1 captures the most variation, PC2 — the second most, and so on. Thus, in this plot above, we can see how 2 of the BM+ samples (79 and 80) are further down the x-axis whereas the 3 BM- samples (81,82, and 83) are further left on the plot, which suggests that there are some key differences amongst the two groups.

pcaExplorer

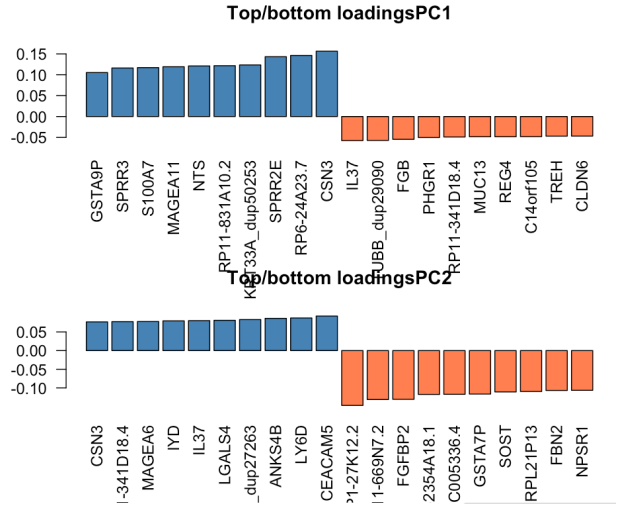
```
#BiocManager::install("pcaExplorer")
#pcaExplorer::pcaExplorer(dds = DESeq.ds, dst = DESeq.rlog)
```

Below are some findings from the pcaExplorer window:

1) The scree plot, shown below, is a plot that demonstrates the cumulative proportion of variance explained as you keep more features. If you calculate the proportion of variance explained for each feature and sort features by proportion of variance explained, you can create this plot and identify the point where adding a new feature has a significant drop in variance.

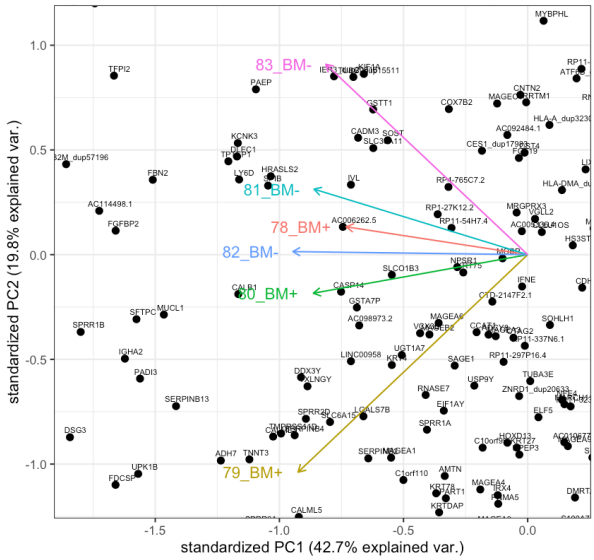


2) The hi loadings plot, shown below, extracts and plots the genes with the highest loadings so this plot shows the top 10 genes that have the greatest loading values in PC1 and PC2. I will further explore these genes later.



3) The genespca plot, shown below, computes and plots the principal components of the genes and displays the samples as in a typical biplot visualization. This provides similar results to the initial PCA plot, where the BM- seem to be pointing in one general direction whereas two of the BM+ samples are pointing downwards. Again, this shows some grouping and potential differences in gene expression.

Zoomed window



Differential Gene Expression Analysis

Running the DE analysis

I wanted to make sure that my conditions were "positive" and "negative" referring to those that have metastasized and those that haven't. It also has to stored as characters, not factors.

```
colData(DESeq.ds)["condition"]  <- as.factor(c("positive", "positive", "positive", "negative", "negative
colData(DESeq.ds)
```

```
## DataFrame with 6 rows and 2 columns
##        condition        sizeFactor
##         <factor>         <numeric>
## 78_BM+  positive  1.10613178488822
## 79_BM+  positive  0.81753482134662
## 80_BM+  positive  1.07075417092073
## 81_BM-  negative  1.07631125569893
## 82_BM-  negative 0.830368019541507
## 83_BM-  negative  1.16546295004044
```

After running DESeq on our object, it is important to adjust the p-values in the light of our numerous hypotheses. Since we performed multiple tests,the number of times that the p-value represents a random event rather than a systematic one become a cause for concern. Since this is RNA-Seq data, the argument has been made that genes with very low read counts can be ignored for downstream since those read counts are considered unreliable and variable.

```
DESeq.ds <- DESeq(DESeq.ds)
DGE.results <- results(DESeq.ds, independentFiltering = TRUE, alpha = 0.05)
summary(DGE.results)
```

```
##
## out of 40876 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)       : 275, 0.67%
```

```
## LFC < 0 (down)     : 201, 0.49%
## outliers [1]       : 1074, 2.6%
## low counts [2]     : 17435, 43%
## (mean count < 7)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
# the DESeqResult object can basically be handled like a data.frame
table(DGE.results$padj < 0.05)
```

```
##
## FALSE  TRUE
## 21891   476
```

Since I only have 3 samples per condition, ~500 genes with differential expression is definitely a good find. From the original paper, they claimed to have found a total of 566 differentially-expressed genes between BM+ and BM- samples that were identified by DESeq2, which is similar to my finding!

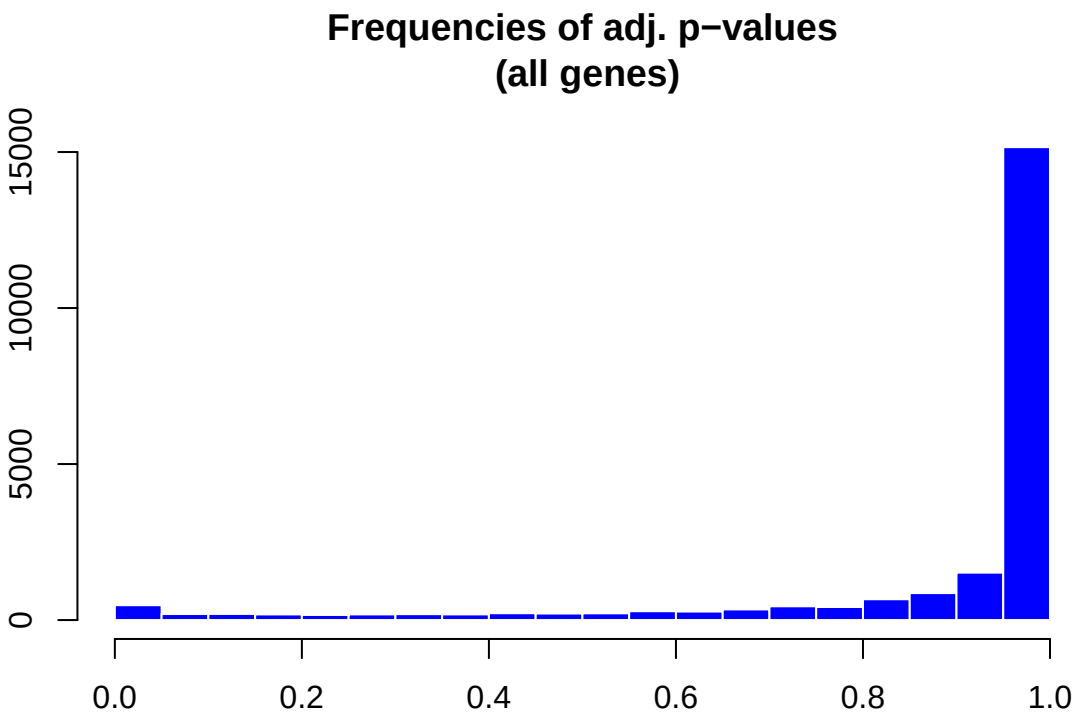
Assessing the DE analysis

Typical questions I would ask myself to make sure my DESeq analysis makes sense:

1) Let's plot the histogram of p-values. I expect a flat distribution along the bottom where most of my pvalues are uniformly distributed between 0 and 1. And I expect a peak close to 0.
2) Individually plot selected genes of interest for which the direction and magnitude of change are known.
3) Plot the heatmaps of lists of genes of interest, which are the most strongly changing ones
4) Produce MA-plots and volcano plots visualizing the magnitude of the change in comparison to the magnitude of baseline expression

Histogram of padj-values

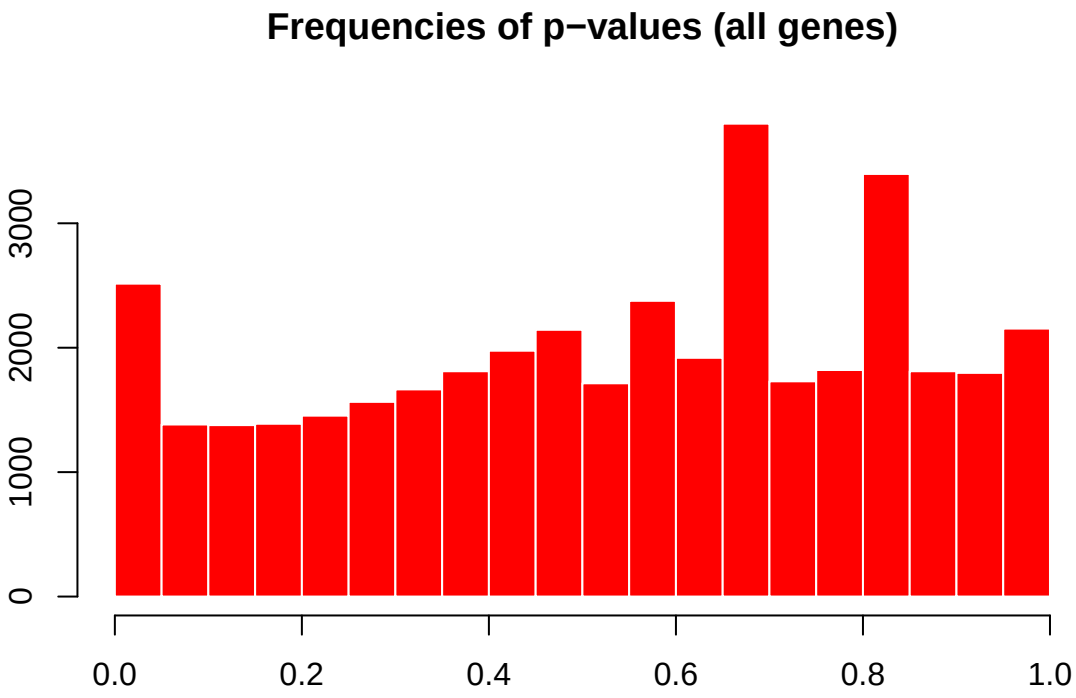
```
hist(DGE.results$padj,
col="blue", border="white", xlab="", ylab="", main="Frequencies of adj. p-values\n (all genes)",
cex = 0.4, breaks = 20)
```



This first was surprising to me because I was expecting flat distribution along the bottom where most of my pvalues are uniformly distributed between 0 and 1 and a peak close to 0. I assumed that the test I was conducting on my data was trying to fit it to a dsitribution that my data was not going to fit well on. However, I realized that the Benjamini-Hochberg transform is nonlinear and pushes a bunch of tests toward 1, so it is hard to read like above. Thus, I plotted the original pvalues to see a uniform distribution.

Histogram of p-values

```
hist(DGE.results$pvalue,
col="red", border="white", xlab="", ylab="", main="Frequencies of p-values (all genes)",
cex = 0.4, breaks = 20)
```



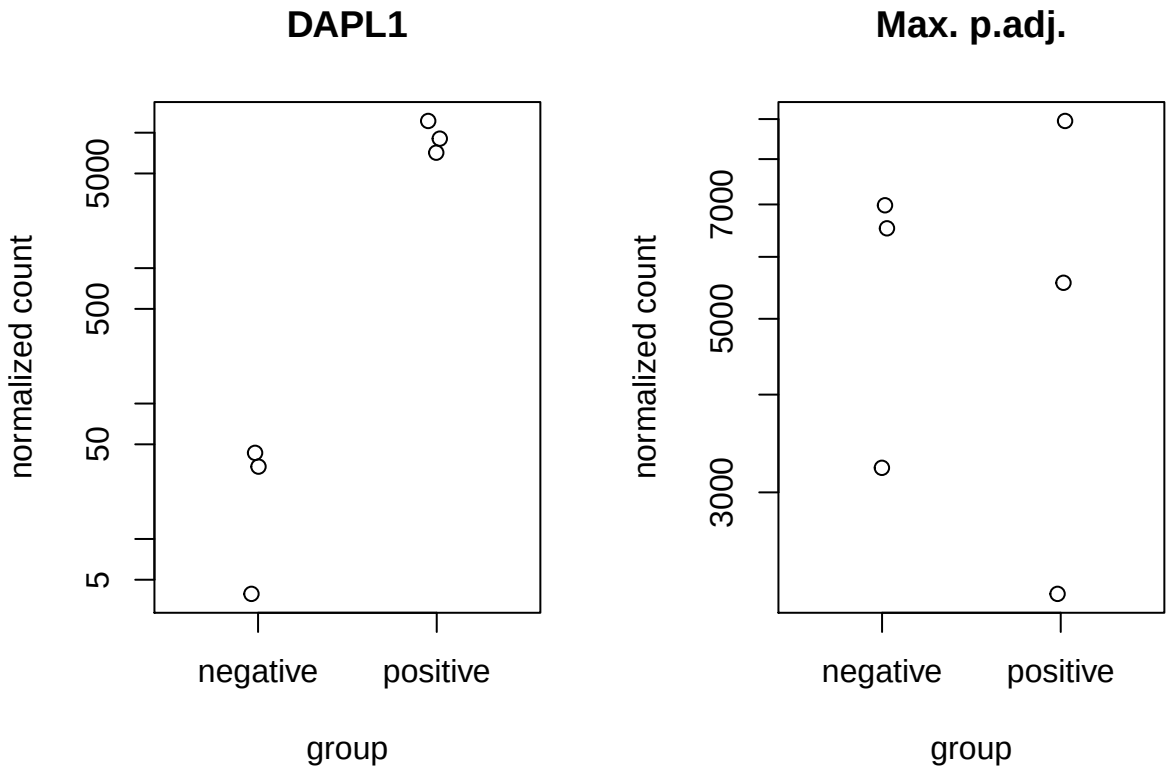

```
DGE.results.sorted <- DGE.results[order(DGE.results$padj),]
head(DGE.results.sorted)
```

```
## log2 fold change (MLE): condition positive vs negative
## Wald test p-value: condition positive vs negative
## DataFrame with 6 rows and 6 columns
##               baseMean    log2FoldChange             lfcSE              stat
##              <numeric>         <numeric>         <numeric>         <numeric>
## DAPL1 4741.18060862026   8.4758601254599 0.921266189383879  9.20022923138897
## DSG3  4928.33306404409  8.95672873037707  1.19394720471306  7.50177955526067
## SYT12 2100.13102183584 -5.38407109224553 0.797262539667327 -6.75319712687384
## PDK4  1134.41528078839 -5.04545962487039 0.766272421398144 -6.58442021920145
## SOX21 369.703864643427  6.53719769418216 0.990479613745796  6.60003255338066
## NTRK2   9411.048254708  6.69344819204477  1.00878752933646  6.63514168979415
##                    pvalue                 padj
##                 <numeric>            <numeric>
## DAPL1  3.5718695692718e-20 7.98920066559024e-16
## DSG3  6.29571884774946e-14 7.04081717338061e-10
## SYT12 1.44622229893331e-11 1.07825513867471e-07
## PDK4  4.56663857293371e-11 1.70236674934681e-07
## SOX21 4.11067509334492e-11 1.70236674934681e-07
## NTRK2 3.24190967308767e-11 1.70236674934681e-07
```

Selected genes of interest

After looking at the list of genes with extremely low p-values, this gene had the lowest adjusted p-value, so I wanted to look at the plot and found that it was very present in one group vs the other since the the DGE analysis was done based on condition.

```
par(mfrow=c(1,2))
plotCounts(DESeq.ds, gene="DAPL1", normalized = TRUE)
plotCounts(DESeq.ds, gene = which.max(DGE.results$padj),
main = "Max. p.adj.")
```

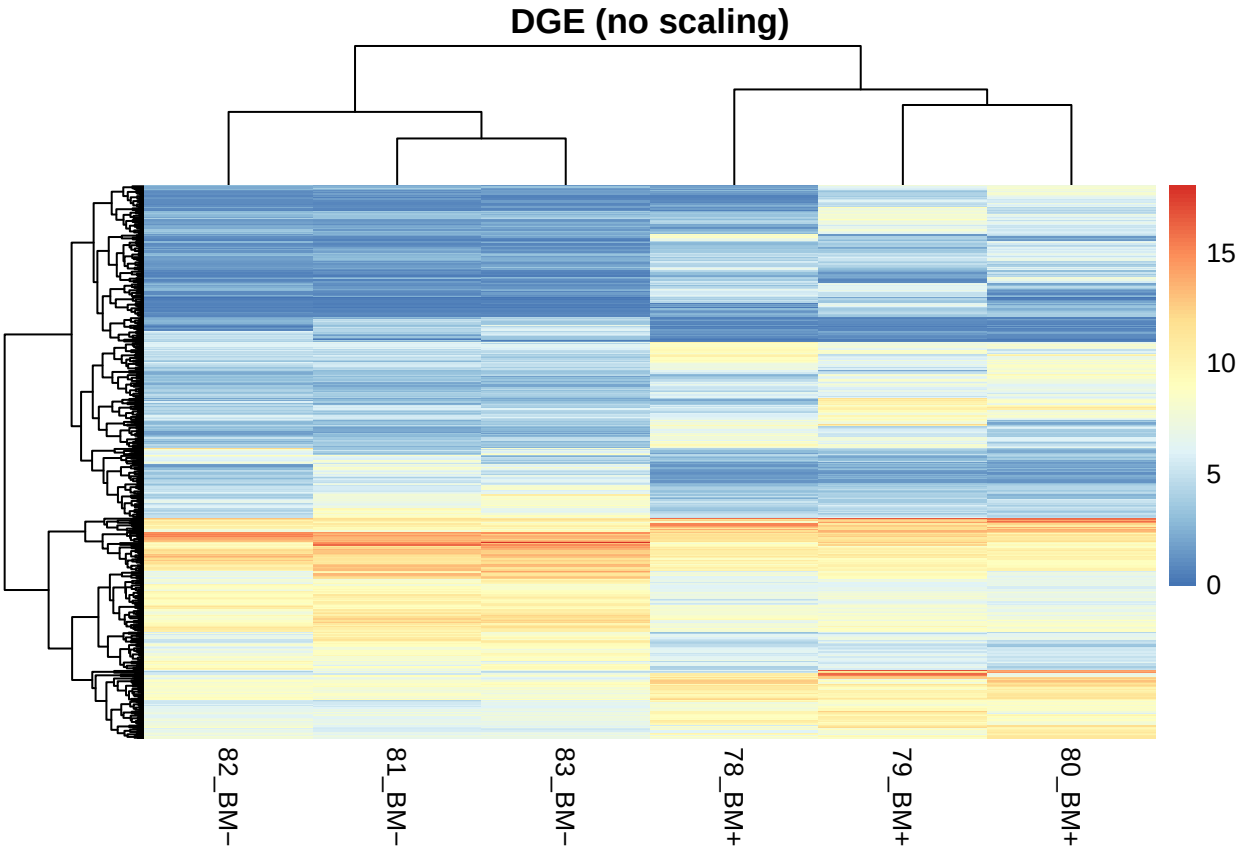


DAPL1 is a very commonly enriched gene in cervical cancer, head and neck cancer, lung cancer, ovarian cancer, or thyroid cancer. This gene is overexpressed in BM+ cells, which is an interesting marker since the gene has very low brain-related location specificity.

Heatmaps of lists of genes of interest

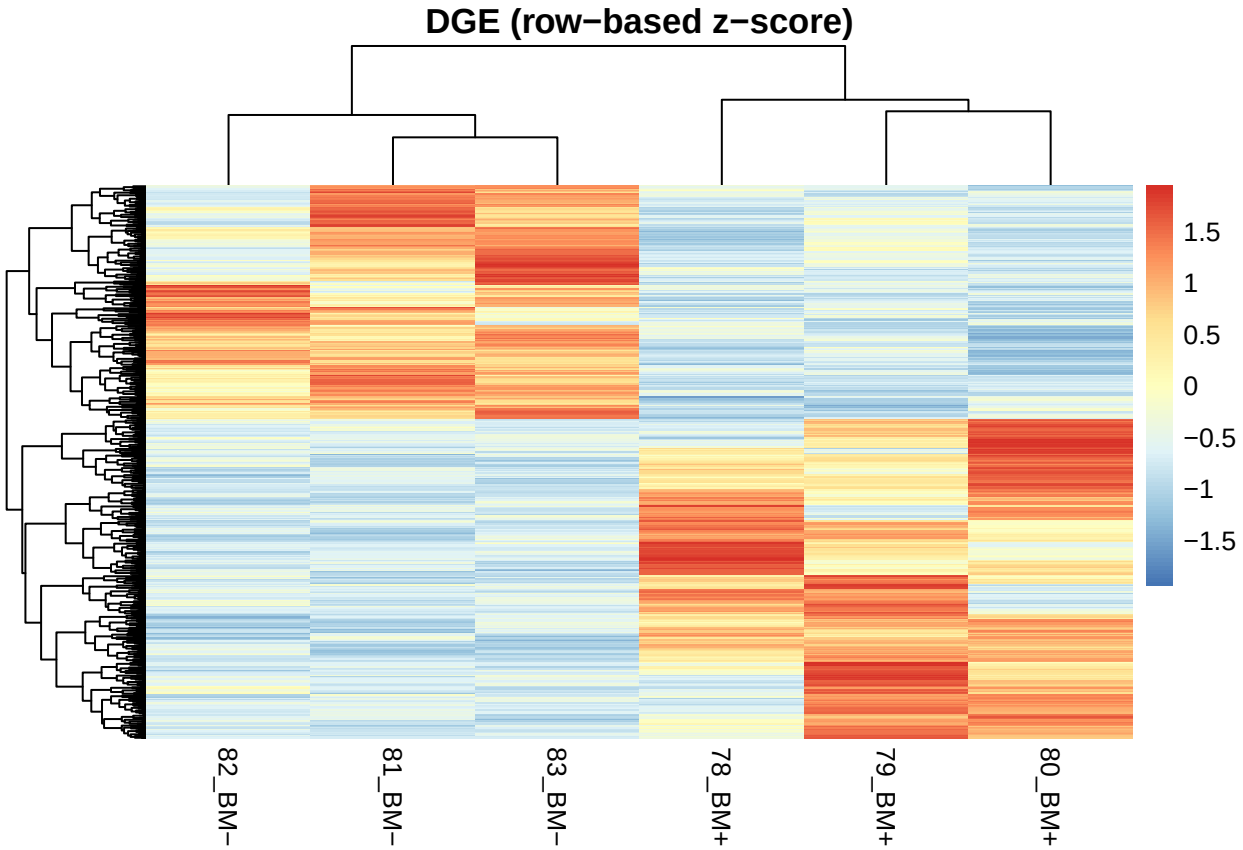
As used earlier in the report, heatmaps have proven useful since it will map all of the genes that show differential expression with adjusted p-value <0.05. The heatmap below begins to show some gene expression present in all samples, which may be genes that are expressed more in lung cells or in lung adenocarcinomas. There are some genes located on the bottom right of the heatmap that suggests that those genes are only active in brain metastasis positive lung cancer samples.

```
# identify genes with the desired adjusted p-value cut-off
DGEgenes <- rownames(subset(DGE.results.sorted, padj < 0.05))
# extract rlog-transformed values into a matrix
rlog.dge <- DESeq.rlog[DGEgenes,] %>% assay
library(pheatmap)
# heatmap of DEG sorted by p.adjust
pheatmap(rlog.dge, scale="none",
show_rownames = FALSE, main = "DGE (no scaling)")
```



Z-score is a measure of distance, in standard deviations, from the mean. Also, z-scores are centered and normalized. Thus, plotting a heatmap with row-based z-scores will help meinterpret a color as x standard deviations from the mean and provide me with a more intuitive idea of the relative variation of that value.

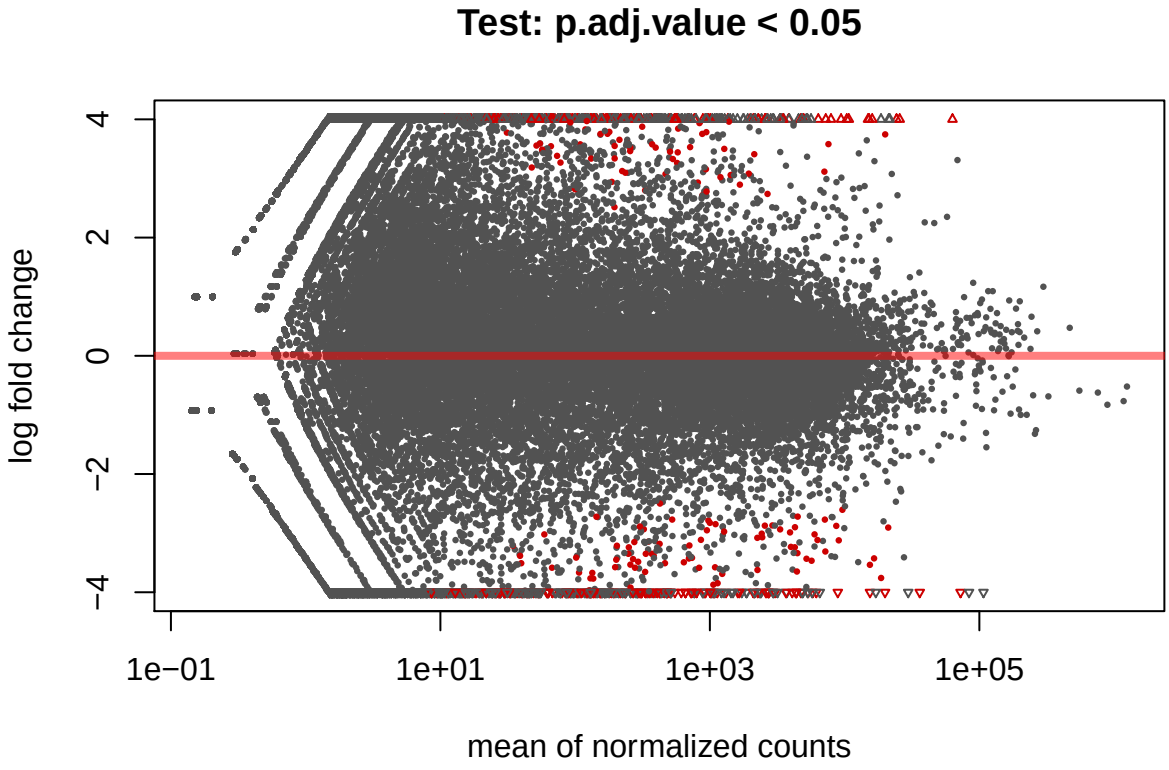
```
pheatmap(rlog.dge, scale="row",
show_rownames = FALSE, main = "DGE (row-based z-score)")
```



MA-plot

An MA plot allows us to look at the relationship between intensity and difference between two data stores for the data. It creates a 2-dimensional plot with a point for each probe. In this plot, the MA-plot provides a global view of the differential genes, with the log2 fold change on the y-axis over the mean of normalized counts. Each dot it a gene and the genes that pass the significance threshold (adjusted p.value <0.05) are colored in red. The x-axis represents the average quantitated value and the y axis shows the difference between them. In my plot, we can see that the majority of my genes are not significant, thus grey. But, at the same time, we can also derive that strongly expressed genes are over-represented in the population of statistically significantly changed genes.

```
plotMA(DGE.results, alpha = 0.05,
main = "Test: p.adj.value < 0.05", ylim = c(-4,4))
```



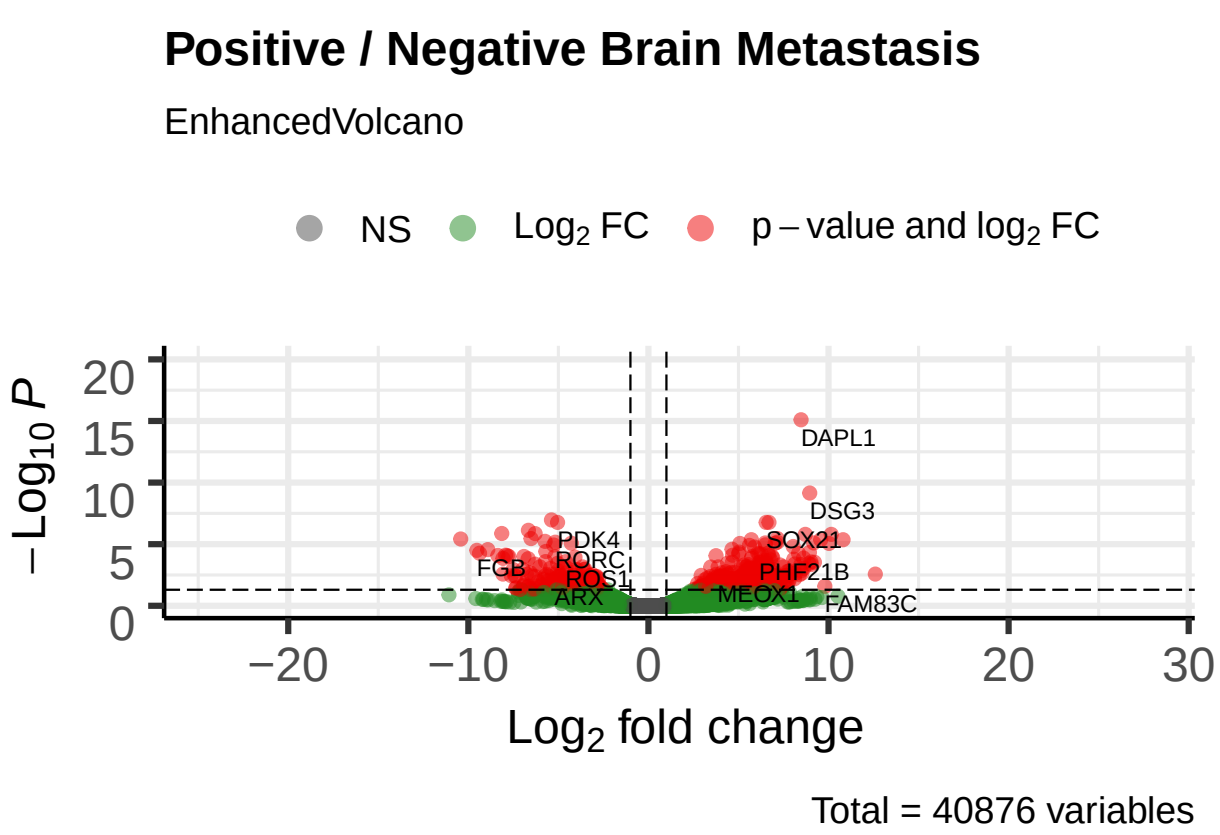
Volcano plots

In a similar fashion, a volcano plot shows the absolute difference seen in a test (x-axis) vs the abs(log10(pvalue)) on the y axis. In the plot, the larger values are more significant. Again, you can see the gene "DAPL1" with a large value and it was one of the most significantly expressed genes with the smallest p-value.

```
#BiocManager::install("EnhancedVolcano")
library(EnhancedVolcano)
```

```
## Loading required package: ggrepel
```

```
vp1 <- EnhancedVolcano(DGE.results, lab = rownames(DGE.results), x = 'log2FoldChange', y = 'padj', pCut
print(vp1)
```



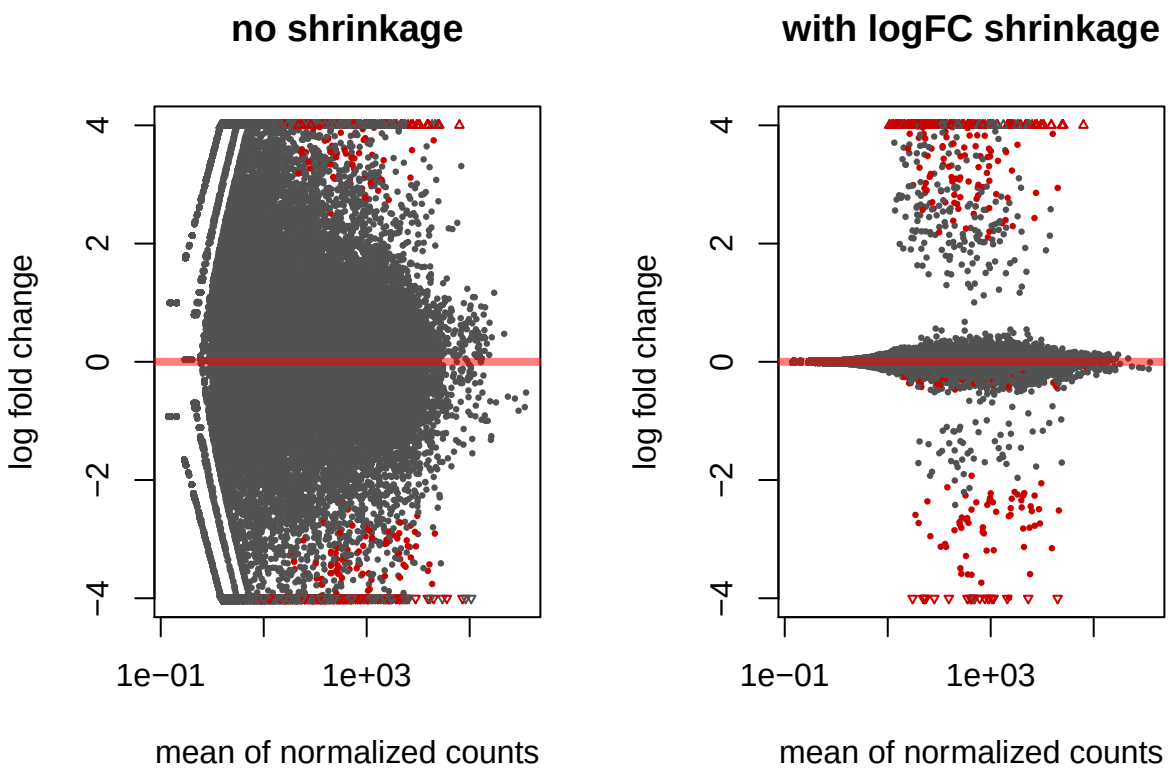
logFC

logFC will reduce the importance of those genes that are lowly and noisily expressed.

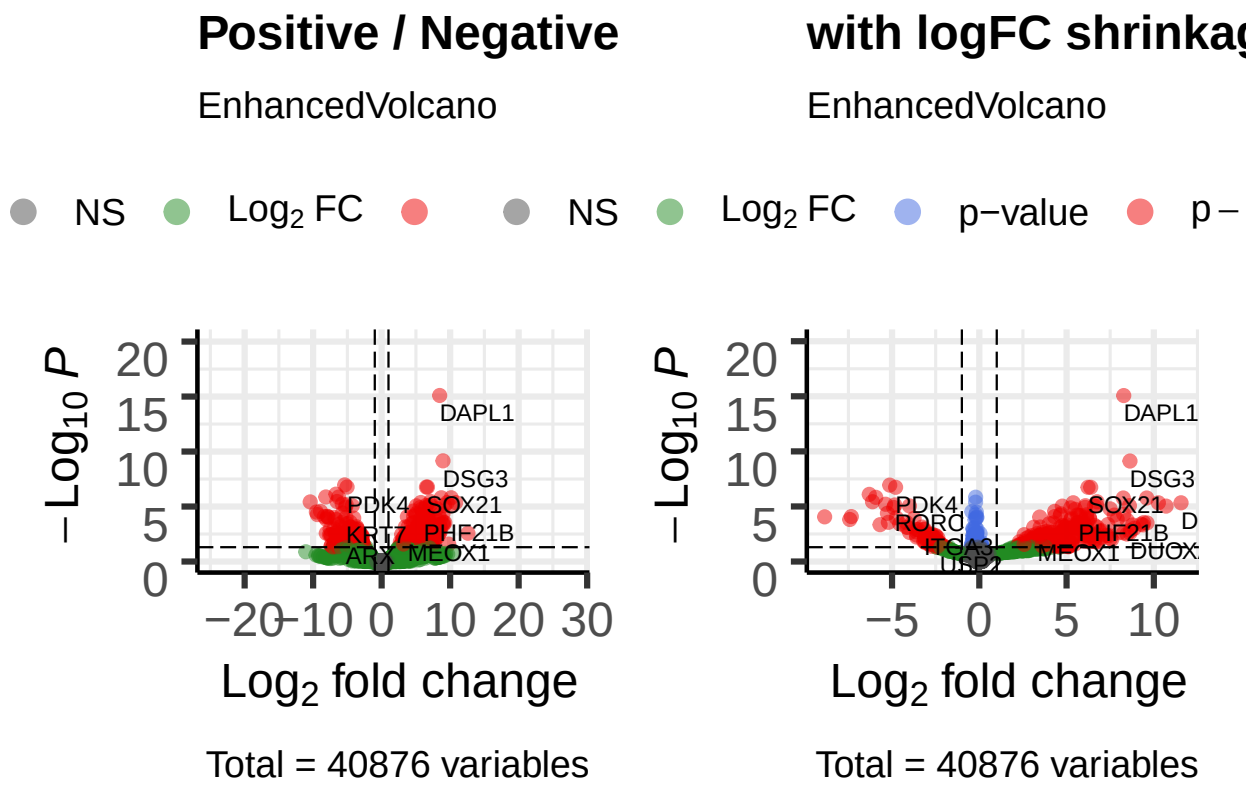
```
DGE.results.shrnk <- lfcShrink(DESeq.ds, coef = 2, type = "apeglm")
```

```
## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

```
par(mfrow = c(1,2))
plotMA(DGE.results, alpha = 0.05, main = "no shrinkage", ylim = c(-4,4))
plotMA(DGE.results.shrnk, alpha = 0.05, main = "with logFC shrinkage", ylim = c(-4,4))
```




```
vp2 <- EnhancedVolcano(DGE.results.shrnk,
lab = rownames(DGE.results.shrnk),
x = 'log2FoldChange',
y = 'padj', pCutoff = 0.05,
title = "with logFC shrinkage")
library(patchwork)
vp1 + vp2
```



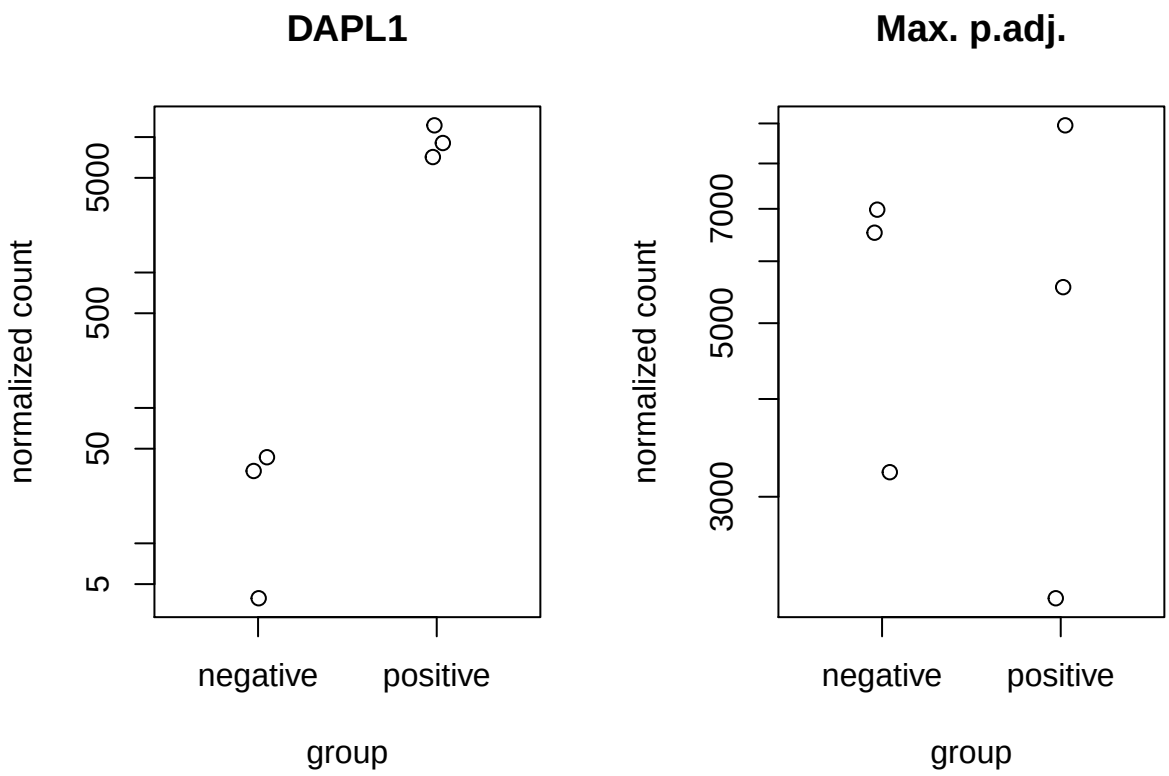
Subset Significant Genes And Analyze

I made a vector of genes for which padj were less than 0.05,thus significant. This vector would have 476 genes (275 upregulated and 201 downregulated in BM+ samples). And I began to observe the top fifteen of those genes which were ordered in significance.

```
DGEgenes <- rownames(subset(DGE.results.sorted, padj < 0.05))
head(DGEgenes, 11)
```

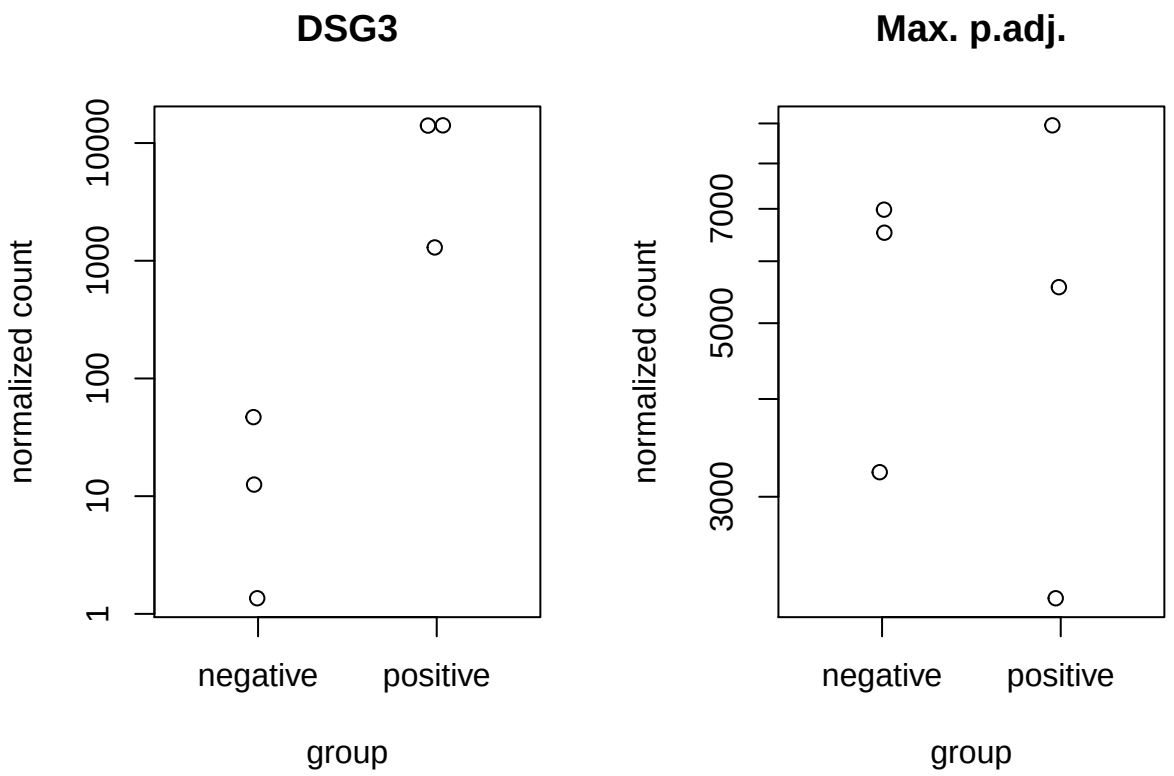
```
##  [1] "DAPL1"          "DSG3"           "SYT12"          "PDK4"
##  [5] "SOX21"          "NTRK2"          "RP3-416H24.1"   "MUC3A"
##  [9] "SLC6A20"        "NTS"            "RP5-1097P24.1"
```

```
par(mfrow=c(1,2))
plotCounts(DESeq.ds, gene="DAPL1", normalized = TRUE)
plotCounts(DESeq.ds, gene = which.max(DGE.results$padj),
main = "Max. p.adj.")
```



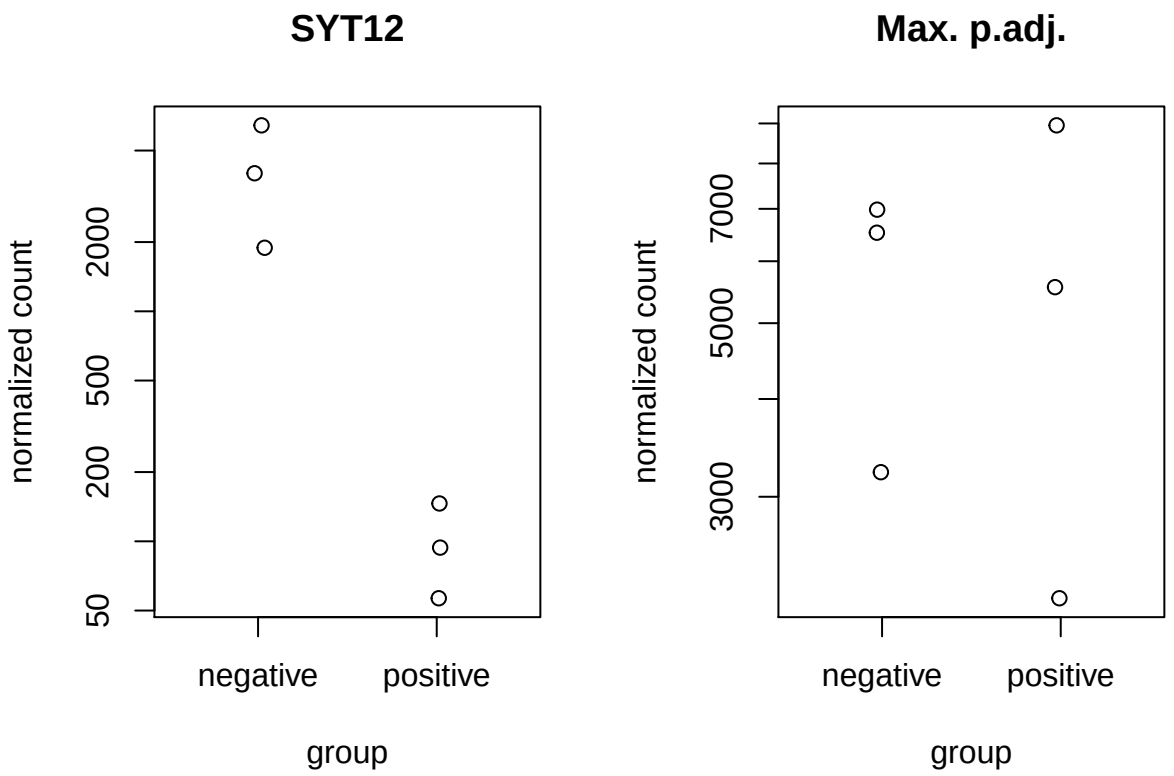
DAPL1 is a very commonly enriched gene in cervical cancer, head and neck cancer, lung cancer, ovarian cancer, or thyroid cancer. This gene is overexpressed in BM+ cells, which is an interesting marker since the gene has very low brain-related location specificity.

```
par(mfrow=c(1,2))
plotCounts(DESeq.ds, gene="DSG3", normalized = TRUE)
plotCounts(DESeq.ds, gene = which.max(DGE.results$padj),
main = "Max. p.adj.")
```



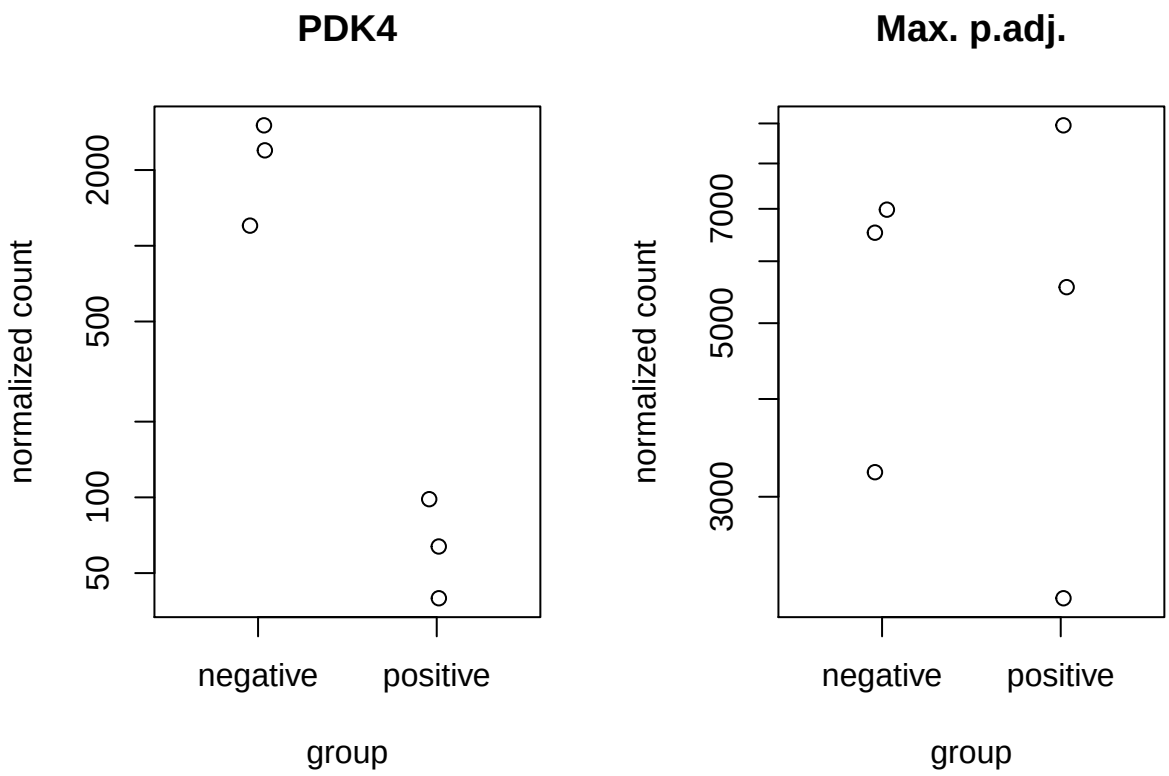
DSG3 is a gene encoding for one of the components of the desmosome, which are button-like points of intercellular contact that allow the attachment of cytoskeletal elements to the plasma membrane at sites of cell-cell. Desmosomes provide strong intercellular adhesion to maintain tissue integrity and homeostasis, which explains why it's higher in BM+ cells and can help facilitate spread. DSG3 has also been shown in several cancers including skin, prostate, lung and head-neck cancer

```
par(mfrow=c(1,2))
plotCounts(DESeq.ds, gene="SYT12", normalized = TRUE)
plotCounts(DESeq.ds, gene = which.max(DGE.results$padj),
main = "Max. p.adj.")
```



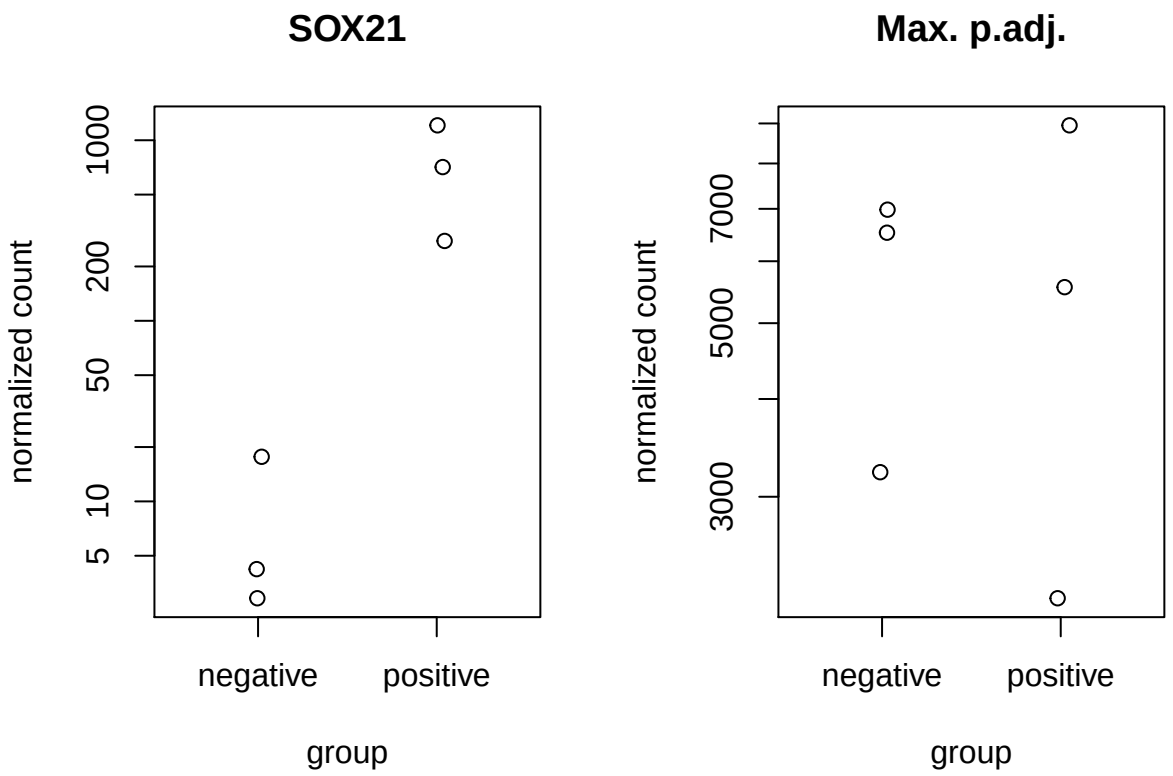
SYT12 codes for a synaptic vesicle phosphoprotein that enhances spontaneous neurotransmitter release but does not effect induced neurotransmitter release. This is downregulated in BM+ samples. The only reason I can see for this is that since this gene mediates calcium-dependent regulation of membrane trafficking in synaptic transmission, brain metasasis positive lung cancer cells don't need to waste energy in communicating with neural cells and thus don't require expression of this gene.

```
par(mfrow=c(1,2))
plotCounts(DESeq.ds, gene="PDK4", normalized = TRUE)
plotCounts(DESeq.ds, gene = which.max(DGE.results$padj),
main = "Max. p.adj.")
```



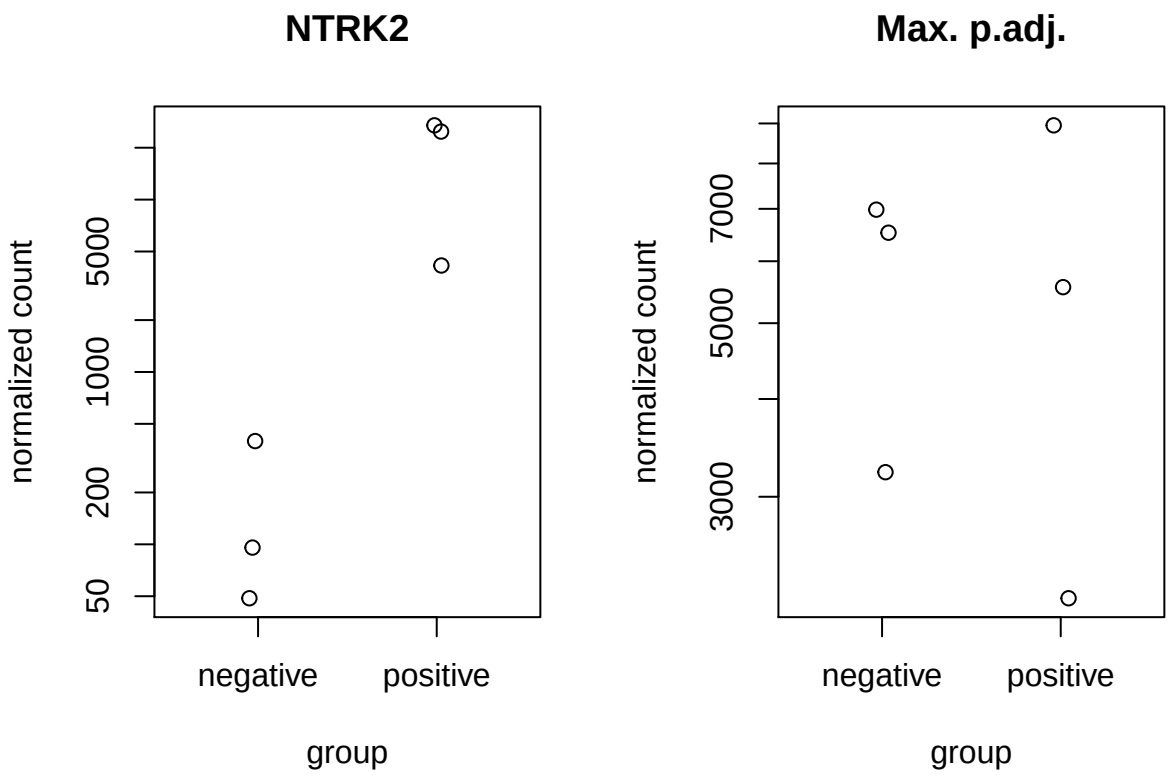
PDK4 plays a role in cell proliferation via its role in regulating carbohydrate and fatty acid metabolism. This is downregulated in BM+ samples, which I was not expecting. Since inhibition of pyruvate dehydrogenase decreases glucose utilization and increases fat metabolism in response to prolonged fasting and starvation, it plays an important role in maintaining normal blood glucose levels under starvation or hypoxic cells, which I imagine cancer cells that would generally not downregulate.

```
par(mfrow=c(1,2))
plotCounts(DESeq.ds, gene="SOX21", normalized = TRUE)
plotCounts(DESeq.ds, gene = which.max(DGE.results$padj),
main = "Max. p.adj.")
```



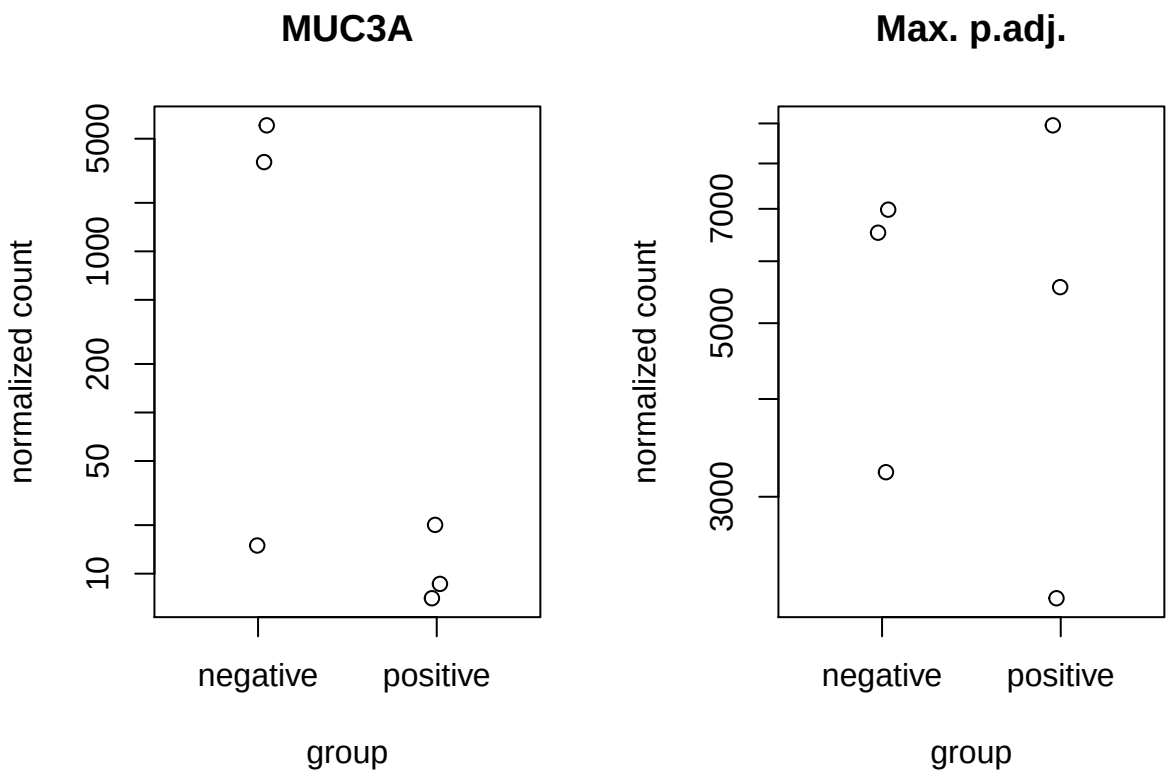
Sox21 promotes neuronal cellular differentiation by counteracting the activity of Sox1, Sox2, and Sox3 as a master regulator and maintains neural cells in an undifferentiated state. This was much higher in the BM+ samples.

```
par(mfrow=c(1,2))
plotCounts(DESeq.ds, gene="NTRK2", normalized = TRUE)
plotCounts(DESeq.ds, gene = which.max(DGE.results$padj),
main = "Max. p.adj.")
```



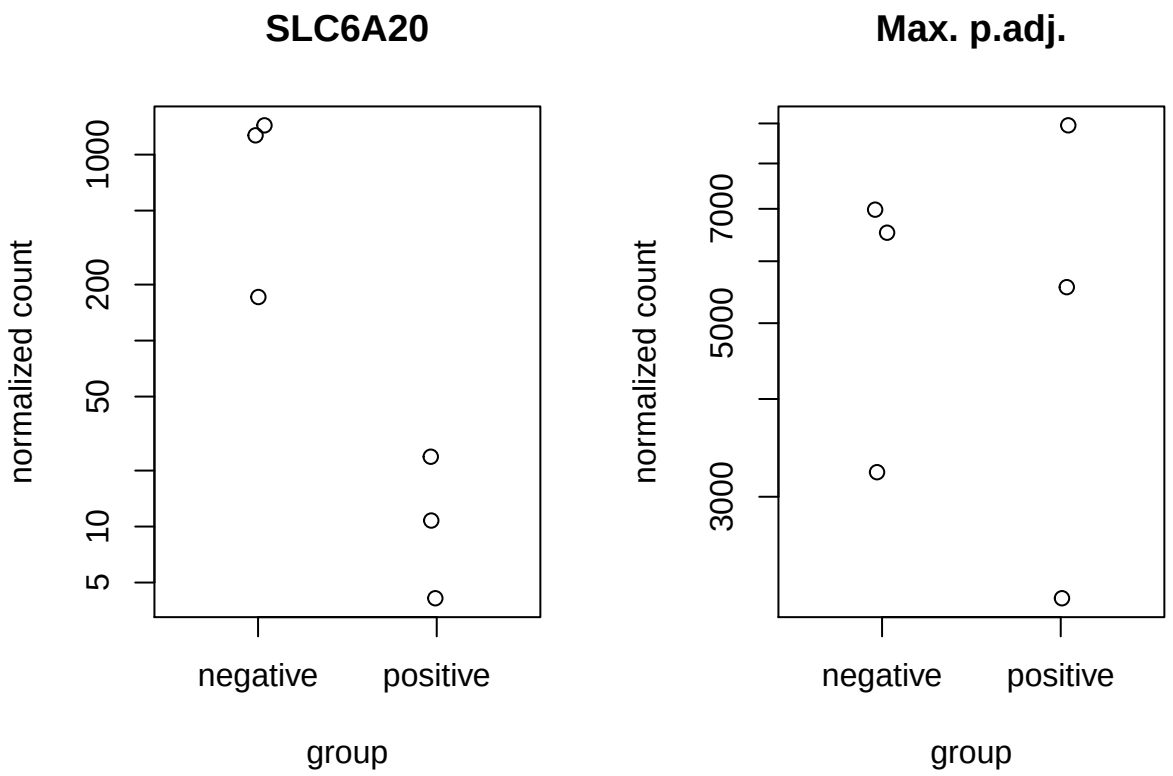
NTRK2 is a receptor tyrosine kinase that is involved in the development and the maturation of the central and the peripheral nervous systems through regulation of neuron survival, proliferation, migration, differentiation, and synapse formation and plasticity. It regulates for instance neuronal differentiation including neurite outgrowth. This was much higher in the BM+ samples which is very interesting because these are lung cancer cells that spread to the brain and would proliferate there. This can be an important signal to look for when biopsying lung cancer and potentially suggest more distant growths as well.

```
par(mfrow=c(1,2))
plotCounts(DESeq.ds, gene="MUC3A", normalized = TRUE)
plotCounts(DESeq.ds, gene = which.max(DGE.results$padj),
main = "Max. p.adj.")
```



The MUC3A gene encodes epithelial glycoproteins, some of which are secreted and some membrane bound. It is to provide a protective, lubricating barrier against particles and infectious agents at mucosal surfaces and may be involved in ligand binding and intracellular signaling. This, altnough seems to be downregulated, is more variable since both negative and positive counts are low.

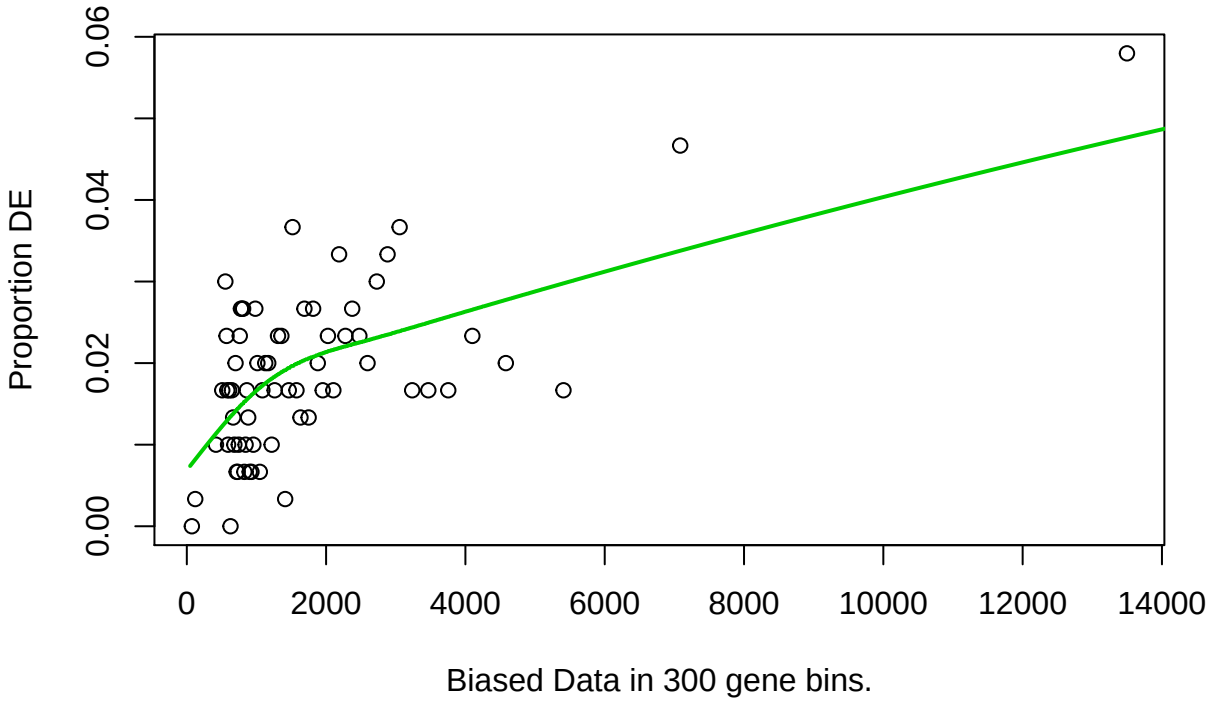
```
par(mfrow=c(1,2))
plotCounts(DESeq.ds, gene="SLC6A20", normalized = TRUE)
plotCounts(DESeq.ds, gene = which.max(DGE.results$padj),
main = "Max. p.adj.")
```



The protein encoded by the SLC6A20 gene is a member of the subgroup of transporter with unidentified substrates within the Na+ and Cl- coupled transporter family, which can help transport of small hydrophilic substances across cell membranes. This is strangely downregulated in BM+ samples.

GO Enrichment Analysis

```
library(goseq) # package for GO term enrichment

# Constructing a named vector of 0 (= not DE) and 1 (= DEG)
gene.vector <- row.names(DGE.results.shrnk) %in% DGEgenes %>% as.integer
names(gene.vector) <- row.names(DGE.results.shrnk)

#TxDb.Hsapiens.UCSC.hg38.knownGene
pwf <- nullp(gene.vector, "hg38", "geneSymbol")
```



PWF (a probability weighting function) quantifies how the probability of a gene selected as DE changes as a function of its transcript length, which is plotted here. Each gene is assigned a binary value (zero or one) according to whether or not the gene is found to be DE, which is determined in the gene.vector. Although this plot doesn't exactly resemble the usual output of the pwf, it still gives insight into our data. Each point represents the percentage of genes called DE in a bin of 300 genes plotted against the median gene length for the bin. The green line is the probability weighting function obtained by fitting to the data. I believe the accurate interpretation of this data is that there are more genes with shorter gene lengths in my data set.

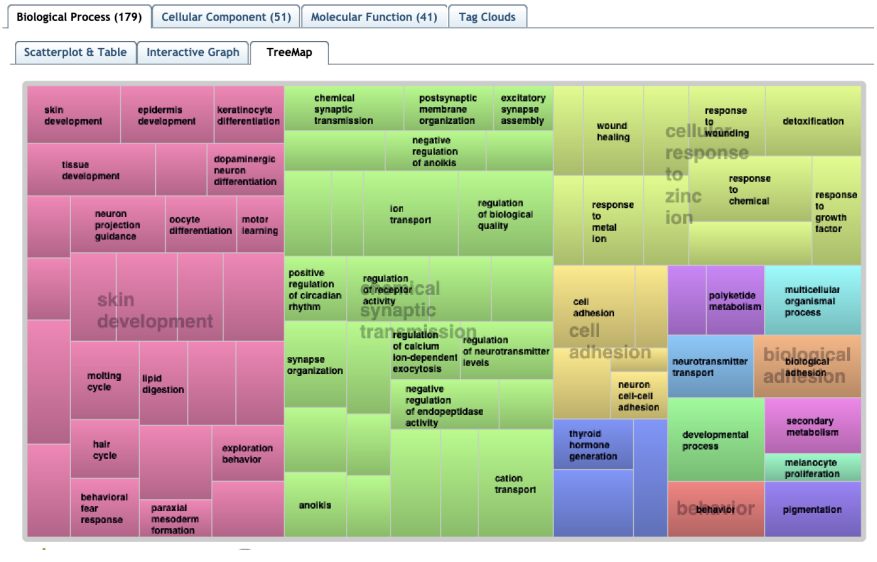
```
#Runa GO
GO.wall <- goseq(pwf, "hg38", "geneSymbol")
go_gns <- getgo( rownames(DGE.results.shrnk), 'hg38', 'geneSymbol') %>% stack

# in principle, the gene information could be added to the goseq results:
merge(GO.wall, go_gns, by.x = "category", by.y = "values") %>% dim
```

```
## [1] 2041868       8
```

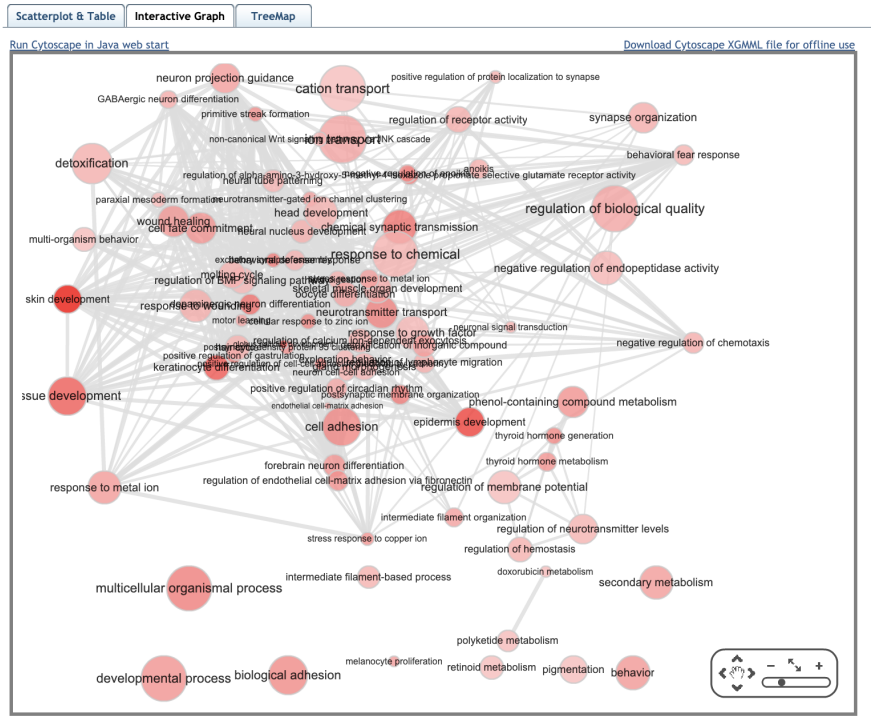
```
subset(GO.wall, over_represented_pvalue < 0.01, select = c("category","over_represented_pvalue")) %>% w
```

More than 40% of gene names specified did not match the gene names for genome hg38 and ID geneSymbol, which means that a good chunk of my data was lost here. However, I still took the txt file produced to http://revigo.irb.hr and ran REVIGO. Using 60% of my cells moving forward, the following were some of my plots that provided a lot of insight about the biological processes:

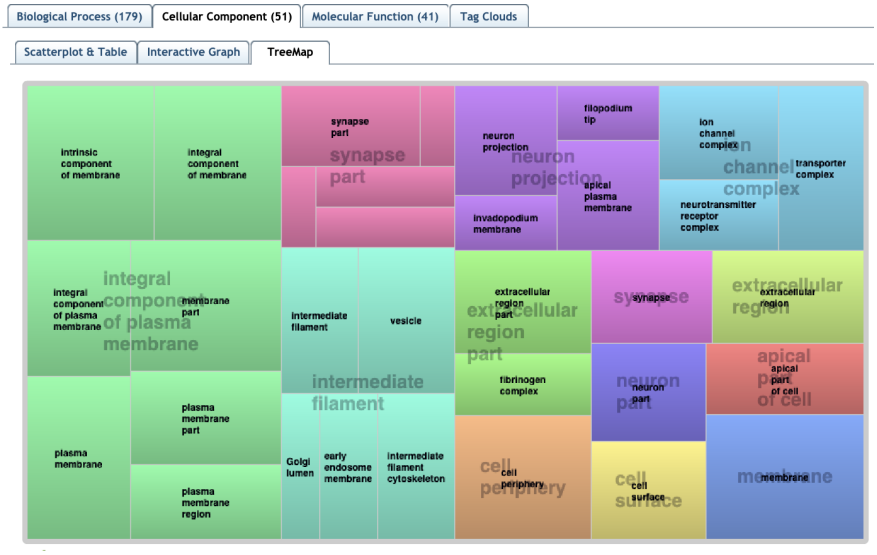
The scatterplot shows the cluster representatives in a two dimensional space. The bubble color indicates the user-provided p-value and their size indicates the frequency of the GO term.

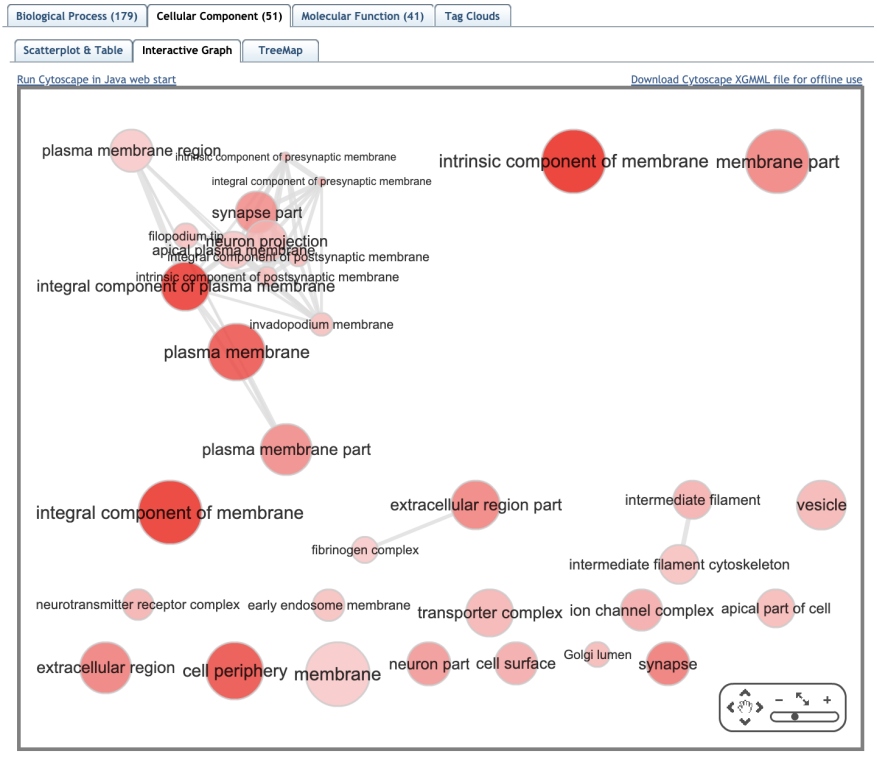


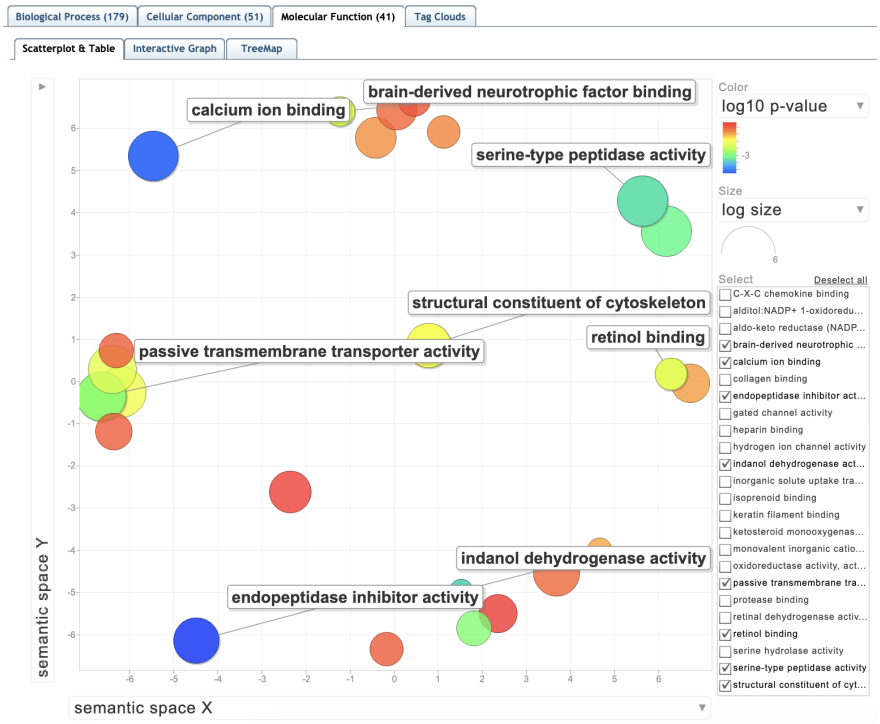
The treemap plot shows multiple rectangles, each of which is a single cluster representative. The representatives are joined into 'superclusters' of loosely related terms and visualized with different colors. In this diagram, the size of the rectangles depict the frequency of the GO term.

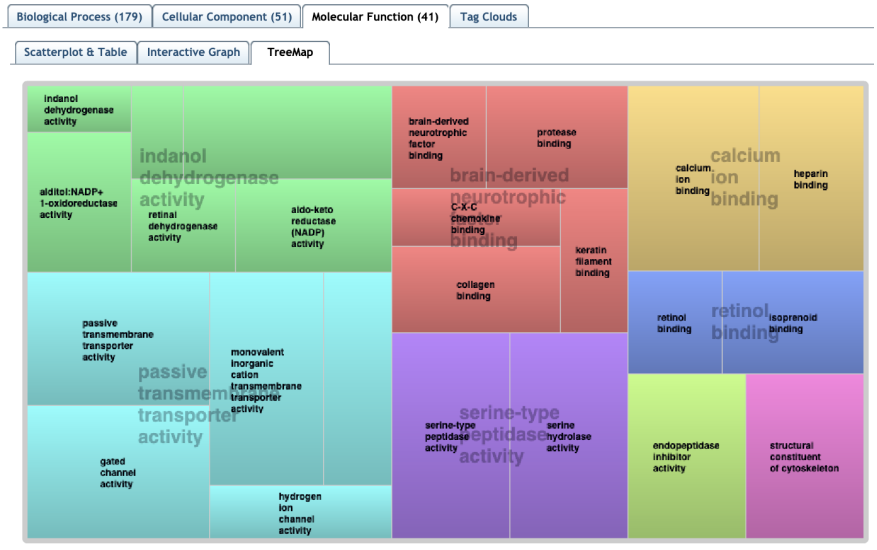


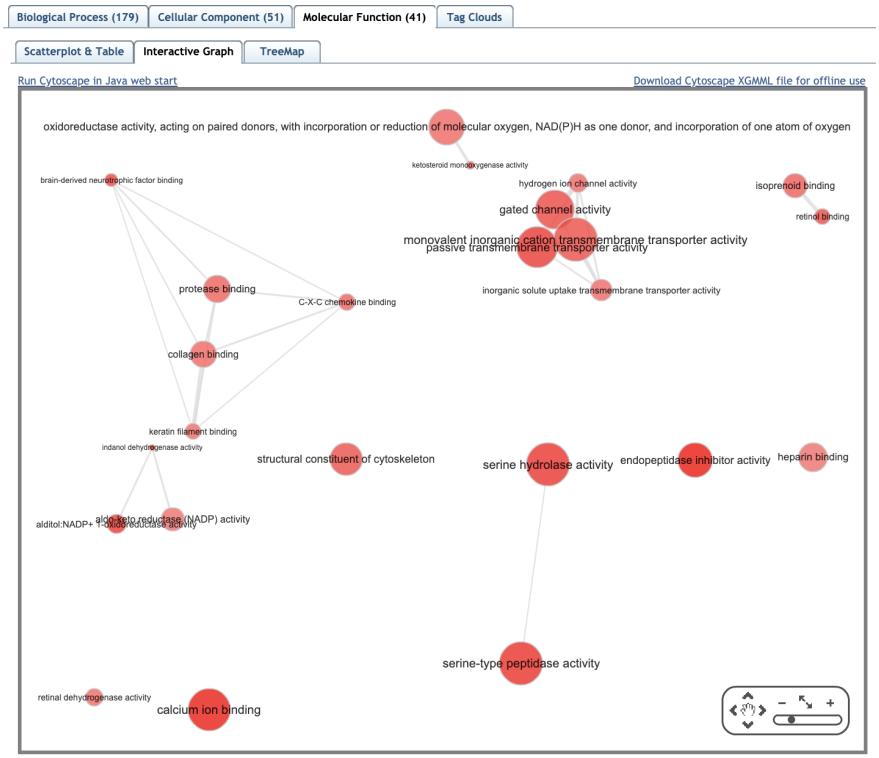
In the network map, the color indicates the p-value I provided and the bubble size indicates the frequency of the GO term. GO terms that are highly similar are linked by edges in the graph whereas the line width indicates the degree of similarity. However, I did move the bubbles around as I was observing the network.


Biological Process (179)
Cellular Component (51)
Molecular Function (41)
Tag Clouds
Scatterplot & Table
Interactive Graph
TreeMap
intrinsic component of membrane
integral component of membrane
synapse part
neuron projection
filopodium tip
apical plasma membrane
invadopodium membrane
ion channel complex
transporter complex
neurotransmitter receptor complex
integral component of plasma membrane
plasma membrane part
plasma membrane
plasma membrane region
intermediate filament
vesicle
Golgi lumen
early endosome membrane
intermediate filament cytoskeleton
extracellular region part
fibrinogen complex
cell periphery
synapse
extracellular region
neuron part
apical part of cell
cell surface
membrane


Biological Process (179)
Cellular Component (51)
Molecular Function (41)
Tag Clouds
Scatterplot & Table
Interactive Graph
TreeMap
Run Cytoscape in Java web start
Download Cytoscape XGMML file for offline use
plasma membrane region
intrinsic component of presynaptic membrane
integral component of presynaptic membrane
synapse part
filopodium tip
neuron projection
apical plasma membrane
integral component of postsynaptic membrane
intrinsic component of postsynaptic membrane
integral component of plasma membrane
invadopodium membrane
plasma membrane
plasma membrane part
intrinsic component of membrane
membrane part
integral component of membrane
extracellular region part
fibrinogen complex
intermediate filament
vesicle
intermediate filament cytoskeleton
neurotransmitter receptor complex
early endosome membrane
transporter complex
ion channel complex
apical part of cell
extracellular region
cell periphery
membrane
neuron part
cell surface
Golgi lumen
synapse


Biological Process (179)
Cellular Component (51)
Molecular Function (41)
Tag Clouds
Scatterplot & Table
Interactive Graph
TreeMap
brain-derived neurotrophic factor binding
calcium ion binding
serine-type peptidase activity
structural constituent of cytoskeleton
retinol binding
passive transmembrane transporter activity
indanol dehydrogenase activity
endopeptidase inhibitor activity
semantic space Y
semantic space X
Color
log10 p-value
Size
log size
Select
Deselect all
C-X-C chemokine binding
alditol:NADP+ 1-oxidoredu...
aldo-keto reductase (NADP...
brain-derived neurotrophic ...
calcium ion binding
collagen binding
endopeptidase inhibitor act...
gated channel activity
heparin binding
hydrogen ion channel activity
indanol dehydrogenase act...
inorganic solute uptake tra...
isoprenoid binding
keratin filament binding
ketosteroid monooxygenas...
monovalent inorganic catio...
oxidoreductase activity, act...
passive transmembrane tra...
protease binding
retinal dehydrogenase activ...
retinol binding
serine hydrolase activity
serine-type peptidase activity
structural constituent of cyt...


Biological Process (179)
Cellular Component (51)
Molecular Function (41)
Tag Clouds
Scatterplot & Table
Interactive Graph
TreeMap
indanol dehydrogenase activity
indanol dehydrogenase activity
alditol:NADP+ 1-oxidoreductase activity
retinal dehydrogenase activity
aldo-keto reductase (NADP) activity
brain-derived neurotrophic factor binding
protease binding
brain-derived neurotrophic binding
C-X-C chemokine binding
keratin filament binding
collagen binding
calcium ion binding
calcium ion binding
heparin binding
retinol binding
retinol binding
isoprenoid binding
passive transmembrane transporter activity
passive transmembrane transporter activity
monovalent inorganic cation transmembrane transporter activity
gated channel activity
hydrogen ion channel activity
serine-type peptidase activity
serine-type peptidase activity
serine hydrolase activity
endopeptidase inhibitor activity
structural constituent of cytoskeleton



By observing the REVIGO output, the genes were involved heavily in the following biological processes: skin development, synaptic transmission, cellular response to the zinc ion, cell adhesion, and biological adhesion. Many of these genes were highly similar in the biological processes, which is expected because many genes work in unison and can cause cascades of one another. Regarding cellular components, I observed a trend of many genes being involved in the membrane and synapse region of the cells, which suggests that many genes played an integral part in the plasma membrane. Finally, for molecular function, many genes were involved in transmembrane transport activity and neurotrophic factor binding.

Key Data Sets

Key data sets that I have generated during the analyses and decided to keep:

1) readcounts, readcounts_sym: these are original data sets of all of my featureCounts with and with out the gene symbols
2) orig_names: keep the names of the read counts
3) mat.counts: keep track of which genes I was missing (when converting from Ensembl to Gene Symbol)
4) symbols_unique: keep track of which unique genes I had
5) DESeq.ds: DESeq object
6) log.norm.counts: sequencing depth normalized log2(read counts)
7) DESeq.rlog: a different kind of object to keep rlog-transformed data
8) rlog.norm.counts: store DESeq.rlog data using function, assay
9) SV_FinalProject.RData: to make exploratory plots
10) corr_coeff: after using Pearson statistic
11) DGE.results.sorted: sorted by padj to order genes in significance of expression
12) gene.vector : for GO analysis
13) Enriched_GOterms_goseq.txt: for REVIGO analysis

Discussion

First and foremost, I was able to replicate the data analysis pipeline to find approximately 476 differentially expressed Gene symbols that differed between brain metastasis positive and negative lung cancer samples. I thoroughly outlined my methods and results, highlighting what I learned at each step of the way. Beginning with my SRA files, I had to ensure proper conversion into fastq files considering they were paired reads. After running fastqc on my files, I learned that it would be beneficial for me to trim my files despite only ~2% adapter sequence presence in the fastqc output. If I were to repeat this project, I would either not trim my sequences or set a minimum overlap of ~3 bases for trimming so I wouldn't lose all of my As near the end. Once, I confirmed quality of sequencing, I continued with aligning.

Using STAR to align my sequences to the hg38 Human Genome Index, I acheived high mapping percentages for all of my samples (80+%). Some of my other peers were receiving lower percentages of mapping, so I would look into why that discrepancy existed for them and was not visibly present in my data set. Perhaps my SRA files were preprocessed or cleaned in some way, which I would like to learn more about. After alignment came featureCounts, which provided the most obstacles for me regarding EnsemblIDs conversion to GeneSymbols. I lost a chunk of my data which was outlined in the methods because some EnsemblIDs didnt map to GeneSymbols. Also, different EnsemblIDs mapped to the same GeneSymbol. However, instead of losing more data, I gave them unique IDs to be able to continue analysis of the data I did have. In the future, I would like to spend more time looking at different ways to create ID for genes and how those IDs translate to one another and into biological functions.

Finally, after normalizing my gene counts, I was able to get interesting information about those genes that were upregulated and downregulated in BM+ samples. Most of the genes that were upregulated were protein coding for neuron formation or cell-cell adhesion, whereas those that were downregulated were related to hypoxia and cell homeostasis. For further analysis of this question, it would be very interesting to get more patient samples and dig deeper into the REVIGO analysis. Additionally, Hubbs et al. (2010) identified several factors that were associated with a higher risk of developing brain metastases, including younger age, increasing size, lymphovascular space invasion, and hilar lymph-node involvement. It would be interesting to add this information for each patient and see if we can get more information about risks, suggesting that more information about standard clinical and pathological factors may add to identify patients prospectively at highest risk.

References

Bonanno, L., Favaretto, A., Rugge, M., Taron, M., & Rosell, R. (2011). Role of Genotyping in Non-Small Cell Lung Cancer Treatment. Drugs, 71(17), 2231-46.

Castrucci WA, Knisely JP. An update on the treatment of CNS metastases in small cell lung cancer. Cancer J (2008) 14:138–46. doi:10.1097/ PPO.0b013e318172d6e1

Gril B, Evans L, Palmieri D, Steeg PS. Translational research in brain metastasis is identifying molecular pathways that may lead to the development of new therapeutic strategies. Eur J Cancer (2010) 46:1204–10. doi:10.1016/j. ejca.2010.02.033

Hubbs JL, Boyd JA, Hollis D, Chino JP, Saynak M, Kelsey CR. Factors associated with the development of brain metastases: analysis of 975 patients with early stage nonsmall cell lung cancer. Cancer (2010) 116:5038–46. doi:10.1002/ cncr.25254

Lin CY, Chen HJ, Huang CC, Lai LC, Lu TP, Tseng GC, et al. ADAM9 promotes lung cancer metastases to brain by a plasminogen activator-based pathway. Cancer Res (2014) 74:5229–43. doi:10.1158/0008-5472.CAN-13-2995

X, D. (2019, February 19). GEO Accession viewer. Retrieved from https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE126548