

Final_script

Pham Gia Cuong

2023-07-12

Loading the packages

```
require(DESeq2)
```

```
## Loading required package: DESeq2
```

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics
```

```
##  
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:stats':  
##  
## IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':  
##  
## anyDuplicated, append, as.data.frame, basename, cbind, colnames,  
## dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,  
## grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,  
## order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,  
## rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,  
## union, unique, unsplit, which.max, which.min
```

```
##  
## Attaching package: 'S4Vectors'
```

```
## The following objects are masked from 'package:base':  
##  
## expand.grid, I, unname
```

```
## Loading required package: IRanges
```

```
## Loading required package: GenomicRanges
```

```
## Loading required package: GenomeInfoDb
```

```
## Loading required package: SummarizedExperiment
```

```
## Loading required package: MatrixGenerics
```

```
## Loading required package: matrixStats
```

```
##  
## Attaching package: 'MatrixGenerics'
```

```
## The following objects are masked from 'package:matrixStats':  
##  
## colAlls, colAnyNAs, colAnys, colAvgPerRowSet, colCollapse,  
## colCounts, colCummaxs, colCummins, colCumprods, colCumsums,  
## colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,  
## colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,  
## colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,  
## colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,  
## colWeightedMeans, colWeightedMedians, colWeightedSds,  
## colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgPerColSet,  
## rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,  
## rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,  
## rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,  
## rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,  
## rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,  
## rowWeightedMads, rowWeightedMeans, rowWeightedMedians,  
## rowWeightedSds, rowWeightedVars
```

```
## Loading required package: Biobase
```

```
## Welcome to Bioconductor  
##  
## Vignettes contain introductory material; view with  
## 'browseVignettes()'. To cite Bioconductor, see  
## 'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
##  
## Attaching package: 'Biobase'
```

```
## The following object is masked from 'package:MatrixGenerics':  
##  
## rowMedians
```

```
## The following objects are masked from 'package:matrixStats':  
##  
## anyMissing, rowMedians
```

```
require('biomaRt')
```

```
## Loading required package: biomaRt
```

```
require(survival)
```

```
## Loading required package: survival
```

```
require(data.table)
```

```
## Loading required package: data.table
```

```
##  
## Attaching package: 'data.table'
```

```
## The following object is masked from 'package:SummarizedExperiment':  
##  
##      shift
```

```
## The following object is masked from 'package:GenomicRanges':  
##  
##      shift
```

```
## The following object is masked from 'package:IRanges':  
##  
##      shift
```

```
## The following objects are masked from 'package:S4Vectors':  
##  
##      first, second
```

```
require(data.table)  
require(dplyr)
```

```
## Loading required package: dplyr
```

```
##  
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:data.table':  
##  
##      between, first, last
```

```
## The following object is masked from 'package:biomaRt':  
##  
##   select
```

```
## The following object is masked from 'package:Biobase':  
##  
##   combine
```

```
## The following object is masked from 'package:matrixStats':  
##  
##   count
```

```
## The following objects are masked from 'package:GenomicRanges':  
##  
##   intersect, setdiff, union
```

```
## The following object is masked from 'package:GenomeInfoDb':  
##  
##   intersect
```

```
## The following objects are masked from 'package:IRanges':  
##  
##   collapse, desc, intersect, setdiff, slice, union
```

```
## The following objects are masked from 'package:S4Vectors':  
##  
##   first, intersect, rename, setdiff, setequal, union
```

```
## The following objects are masked from 'package:BiocGenerics':  
##  
##   combine, intersect, setdiff, union
```

```
## The following objects are masked from 'package:stats':  
##  
##   filter, lag
```

```
## The following objects are masked from 'package:base':  
##  
##   intersect, setdiff, setequal, union
```

```
require(glmnet)
```

```
## Loading required package: glmnet
```

```
## Loading required package: Matrix
```

```
##  
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:S4Vectors':  
##  
##      expand
```

```
## Loaded glmnet 4.1-7
```

```
require(topGO)
```

```
## Loading required package: topGO
```

```
## Loading required package: graph
```

```
## Loading required package: GO.db
```

```
## Loading required package: AnnotationDbi
```

```
##  
## Attaching package: 'AnnotationDbi'
```

```
## The following object is masked from 'package:dplyr':  
##  
##      select
```

```
##
```

```
## Loading required package: SparseM
```

```
##  
## Attaching package: 'SparseM'
```

```
## The following object is masked from 'package:base':  
##  
##      backsolve
```

```
##  
## groupGOTerms:      GOBPterm, GOMFterm, GOCCterm environments built.
```

```
##  
## Attaching package: 'topGO'
```

```
## The following object is masked from 'package:IRanges':
##
##      members
```

Reading the gene expression data

The data is under htseq count format, hence i need to to convert it to raw count matrix. After this step, we have genes.arr containing names of gene profiles. and gene.exp containing raw count matrix.

```
gene.exp <- read.table("/fast/work/users/phgi10_c/Uni/final_project/final_data/htseq_
count/TCGA-BLCA.htseq_counts.tsv.gz",
                      header = TRUE, sep = "\t", stringsAsFactors = FALSE)
gene.exp <- gene.exp[grepl("ENS", gene.exp$Ensembl_ID), ]
gene.exp <- gene.exp[, order(names(gene.exp))]

genecode.id <- gene.exp[,1]
fixed_names <- sapply(strsplit(genecode.id, ".", fixed=T), function(x) x[1])

gene.exp[,1] <- fixed_names

row.names(gene.exp) <- gene.exp[,1]
gene.exp <- gene.exp[-1]
gene.exp <- 2^gene.exp -1
gene.exp <- subset(gene.exp, !(rowSums(gene.exp < 5) == ncol(gene.exp))) # removing l
ow count genes

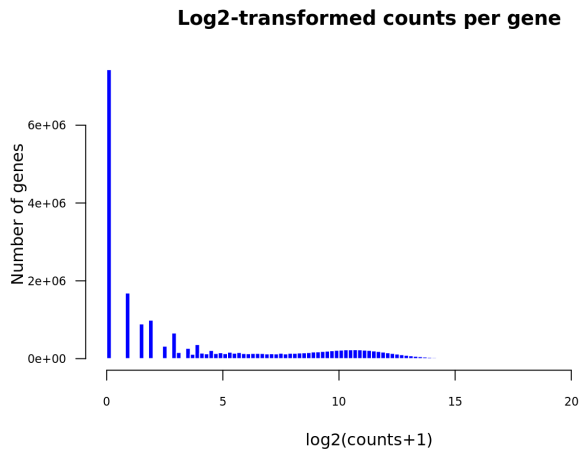
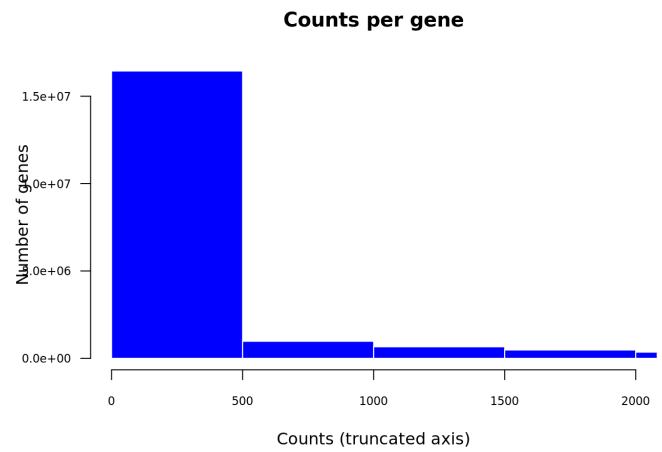
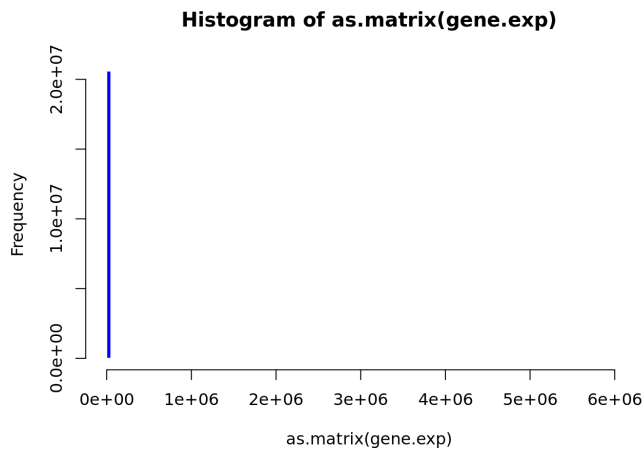
genes.arr <- row.names(gene.exp) #name of genes
write.table(gene.exp, file = "/data/gpfs-1/users/phgi10_c/work/Uni/MultiOmic_Datascie
nce/results/gene_exp_analysis/count_data.csv", sep = "\t", quote = FALSE, row.names =
TRUE)
gene.exp <- as.data.frame(lapply(gene.exp, as.integer))
```

Gene expression data analysis

Ploting the histogram of reads count

From these plots, we can see that after removing low count genes, our data still have any genes, which have smaller than 500 reads.

```
hist(as.matrix(gene.exp), col="blue", border="white", breaks=100)
hist(as.matrix(gene.exp), col="blue", border="white",
     breaks=20000, xlim=c(0,2000), main="Counts per gene",
     xlab="Counts (truncated axis)", ylab="Number of genes",
     las=1, cex.axis=0.7)
epsilon <- 1 # pseudo-count to avoid problems with log(0)
hist(as.matrix(log2(gene.exp + epsilon)), breaks=100, col="blue", border="white",
     main="Log2-transformed counts per gene", xlab="log2(counts+1)", ylab="Number of
genes",
     las=1, cex.axis=0.7)
```



Reading Clinical data

```
clin.data <- read.csv("/fast/work/users/phgi10_c/Uni/final_project/final_data/TCGA-BL
CA.survival.tsv",
                    header = TRUE, sep = "\t", stringsAsFactors = FALSE)

clin.data <- clin.data[order(clin.data$sample), ]
clin.data$sample <- gsub("\\-", ".", clin.data$sample)
color_sample <- ifelse(clin.data$OS == 1, "red", "blue")

#phenotype
phenotype <- read.csv("/data/gpfs-1/users/phgi10_c/work/Uni/final_project/final_data/
phenotype/TCGA-BLCA.GDC_phenotype.tsv.gz", header=TRUE, sep="\t")
phenotype <- phenotype[order(phenotype$submitter_id.samples), ]
phenotype$submitter_id.samples <- gsub("\\-", ".", phenotype$submitter_id.samples)
```

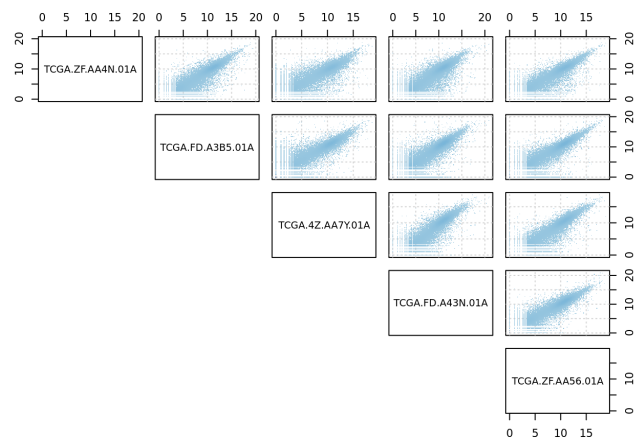
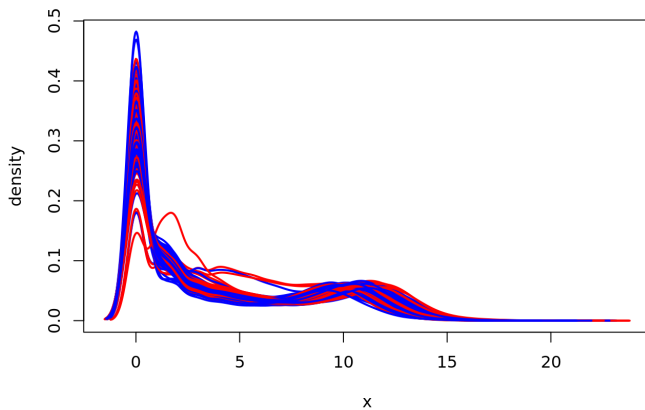
Density plot and correlation pair plot

In the density plot, we can see that the survival and non survival patients have quite similar density in number of read. But those patients also have somewhat different in density of low number of reads. (Red: non survival, Blue: survival)

```
library(affy)
plotDensity(log2((as.matrix(gene.exp) + epsilon)), lty=1, col=color_sample, lwd=2)

# Define a function to draw a scatter plot for a pair of variables (samples) with density colors
plotFun <- function(x,y){
  dns <- densCols(x,y);
  points(x,y, col=dns, pch=".", panel.first=grid());
  # abline(a=0, b=1, col="brown")
}

# Plot the scatter plot for a few pairs of variables selected at random
set.seed(123) # forces the random number generator to produce fixed results
pairs(log2(gene.exp[,sample(ncol(gene.exp),5)] + epsilon),
      panel=plotFun, lower.panel = NULL)
```



Filter samples There are some samples, which are in gene expression data but not in clinical data and phenotype data, hence we need to remove it. After this step, we will have a clinical data including cancer stages. Besides that, gene exp data and clinical data have the same samples.

```
overlapped.samples <- colnames(gene.exp) #samples from transcriptomic data
clin.data <- clin.data[clin.data$sample %in% overlapped.samples, ]
phenotype <- phenotype[phenotype$submitter_id.samples %in% clin.data$sample, ]
phenotype <- phenotype[!grepl("not reported", phenotype$tumor_stage.diagnoses), ]

gene.exp <- gene.exp[,phenotype$submitter_id.samples]
clin.data <- clin.data[clin.data$sample %in% phenotype$submitter_id.samples, ]
phenotype$tumor_stage.diagnoses <- factor(phenotype$tumor_stage.diagnoses, levels = c(
  "stage i", "stage ii", "stage iii", "stage iv"))
phenotype$tumor_stage.diagnoses <- as.numeric(phenotype$tumor_stage.diagnoses)
clin.data$stage <- phenotype$tumor_stage.diagnoses
phenotype <- phenotype[phenotype$lost_follow_up == "N0",]
clin.data <- clin.data[clin.data$sample %in% phenotype$submitter_id.samples, ]
gene.exp <- gene.exp[,phenotype$submitter_id.samples]
write.table(clin.data, file = "/data/gpfs-1/users/phgi10_c/work/Uni/MultiOmic_Datasci
ence/results/gene_exp_analysis/clinical.csv", sep = "\t", quote = FALSE, row.names =
FALSE)
```

Stratifying gene expression data

We removed the stage 1, because all patients survive

```
#separate clinical data by stage
clin_stage_list <- split(clin.data, clin.data$stage)
stage_2 <- clin_stage_list$`2`
stage_3 <- clin_stage_list$`3`
stage_4 <- clin_stage_list$`4`

#separate raw_count data by stage

raw_count_stage2 <- gene.exp[,colnames(gene.exp) %in% stage_2$sample]
raw_count_stage3 <- gene.exp[,colnames(gene.exp) %in% stage_3$sample]
raw_count_stage4 <- gene.exp[,colnames(gene.exp) %in% stage_4$sample]

row.names(raw_count_stage2) <- row.names(gene.exp)
row.names(raw_count_stage3) <- row.names(gene.exp)
row.names(raw_count_stage4) <- row.names(gene.exp)

genes_4 <- row.names(raw_count_stage4)
genes_3 <- row.names(raw_count_stage3)
genes_2 <- row.names(raw_count_stage2)
```

```
DEanalysis <- function(gene.exp, clin.data) {
  dds <- DESeqDataSetFromMatrix(countData=gene.exp,
                                colData=DataFrame(OS=as.factor(clin.data$OS)),
                                design=~OS
                                )
  dds <- estimateSizeFactors(dds)
  dds <- DESeq(dds)
  res <- results(dds)
  return(res)
}
```

```
res_4 <- DEanalysis(raw_count_stage4,stage_4)
```

```
## using pre-existing size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

```
## fitting model and testing
```

```
## -- replacing outliers and refitting for 4696 genes  
## -- DESeq argument 'minReplicatesForReplace' = 7  
## -- original counts are preserved in counts(dds)
```

```
## estimating dispersions
```

```
## fitting model and testing
```

```
res_3 <- DEanalysis(raw_count_stage3,stage_3)
```

```
## using pre-existing size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

```
## fitting model and testing
```

```
## -- replacing outliers and refitting for 5296 genes  
## -- DESeq argument 'minReplicatesForReplace' = 7  
## -- original counts are preserved in counts(dds)
```

```
## estimating dispersions
```

```
## fitting model and testing
```

```
res_2 <- DEanalysis(raw_count_stage2,stage_2)
```

```
## using pre-existing size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

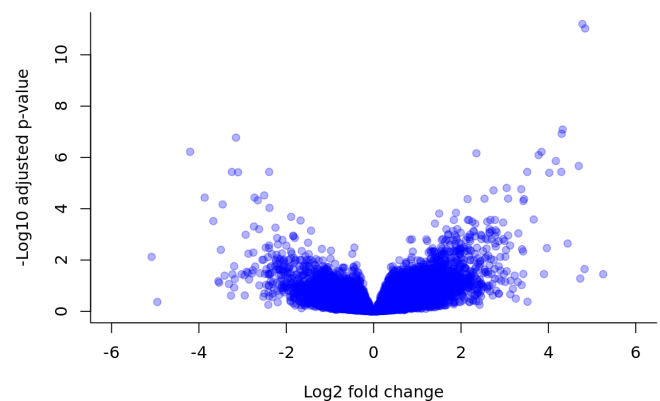
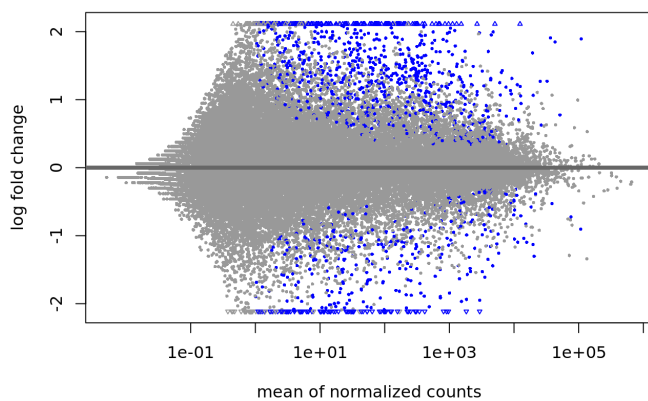
```
## fitting model and testing
```

```
## -- replacing outliers and refitting for 5281 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)
```

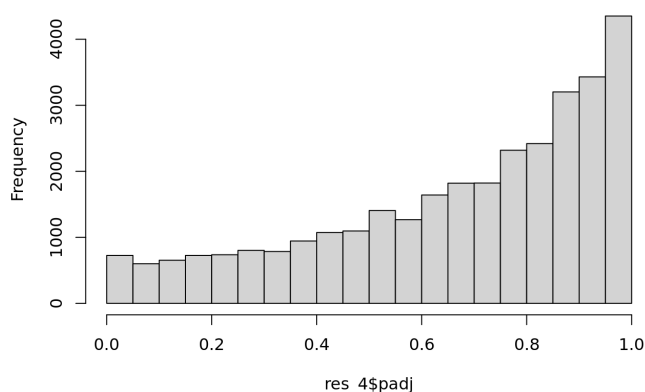
```
## estimating dispersions
```

```
## fitting model and testing
```

```
raw_count_stage4$gene <- genes.arr
plotMA(res_4)
plot(res_4$log2FoldChange, -log10(res_4$padj), pch=19, col=rgb(0,0,1,.3), xlim=c(-6,
6),
      xlab="Log2 fold change", ylab= "-Log10 adjusted p-value", bty="l")
hist(res_4$padj)
```



Histogram of res_4\$padj



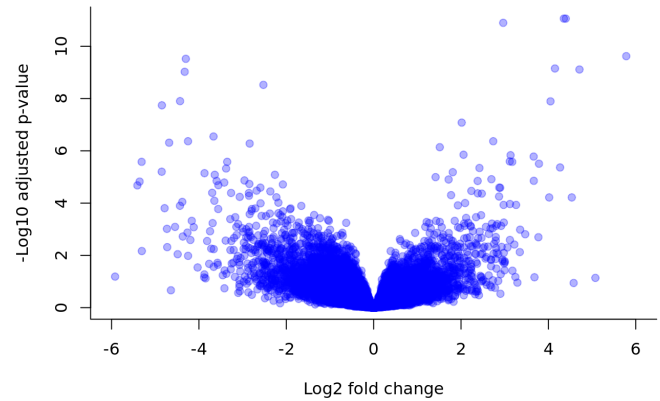
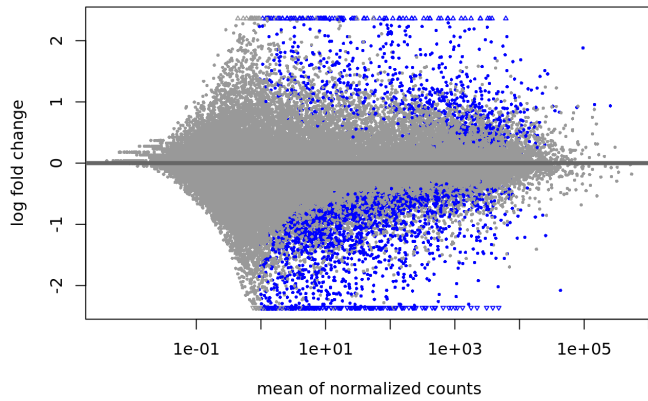
As we can see here that p-value distribution in stage 4

skews on the left side, which means that there is a higher probability of obtaining small p-values than large p-values. In other words, the data provides strong evidence against the null hypothesis, indicating that the observed results are not likely to have occurred by chance alone. One of the main reason here is that our data lacks of non-survival patients in stage 4.

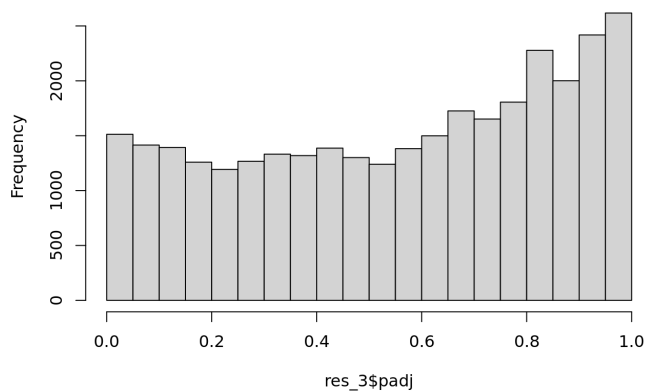
```

raw_count_stage3$gene <- genes.arr
plotMA(res_3)
plot(res_3$log2FoldChange, -log10(res_3$padj), pch=19, col=rgb(0,0,1,.3), xlim=c(-6,
6),
      xlab="Log2 fold change", ylab= "-Log10 adjusted p-value", bty="l")
hist(res_3$padj)

```



Histogram of res_3\$padj



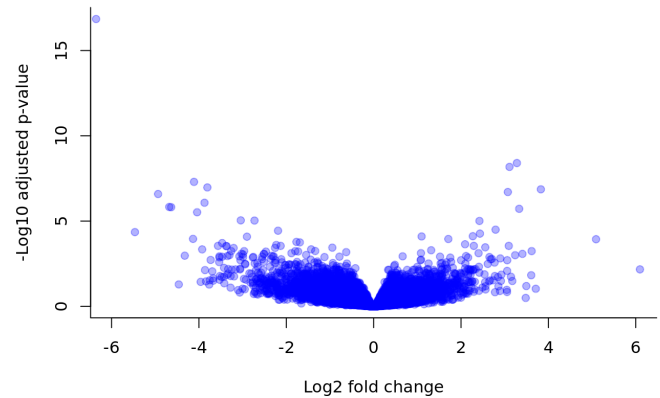
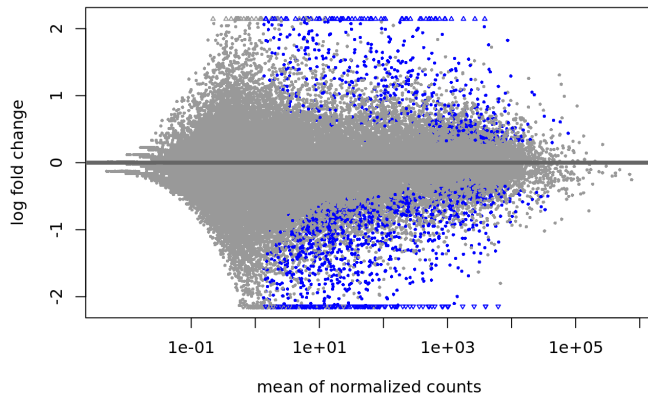
In stage 3, the p-value distribution is somewhat

uniform meaning that the observed data is consistent with what is expected under the null hypothesis. It increases our trustability on this result.

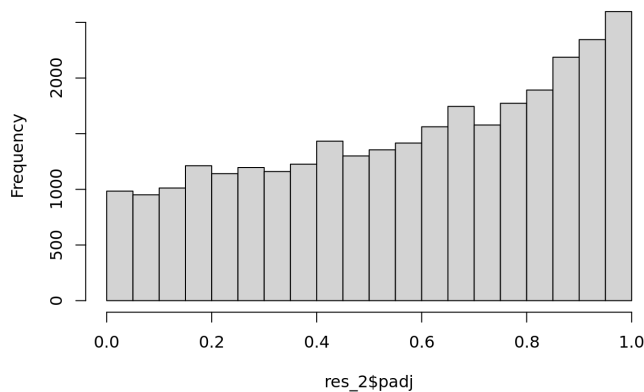
```

raw_count_stage2$gene <- genes.arr
plotMA(res_2)
plot(res_2$log2FoldChange, -log10(res_2$padj), pch=19, col=rgb(0,0,1,.3), xlim=c(-6,
6),
      xlab="Log2 fold change", ylab= "-Log10 adjusted p-value", bty="l")
hist(res_2$padj)

```



Histogram of res_2\$padj



Similar to stage 3, the p-value of stage 2 is also

somewhat uniform distributed. # Getting significant genes in each stage All genes, which have adjusted p-value smaller than 0.05 and absolved log2foldchange greater than 1, are chosen as significant genes.

```
sig_indices_4 <- which(res_4$padj < 0.05 & (res_4$log2FoldChange < -1 | res_4$log2FoldChange > 1))
sig_indices_3 <- which(res_3$padj < 0.05 & (res_3$log2FoldChange < -1 | res_3$log2FoldChange > 1))
sig_indices_2 <- which(res_2$padj < 0.05 & (res_2$log2FoldChange < -1 | res_2$log2FoldChange > 1))

sig_raw_count_4 <- raw_count_stage4[sig_indices_4,]
sig_raw_count_3 <- raw_count_stage3[sig_indices_3,]
sig_raw_count_2 <- raw_count_stage2[sig_indices_2,]
```

Getting genes which contains CNV

In this step, we find the genes, which have independent and recurrent CNV, from genes profile of gene expression data.

```

cnv_final <- read.csv("/data/gpfs-1/users/phgi10_c/work/Uni/final_project/cnv_analysis/final_cnv.csv",stringsAsFactors=FALSE, header=TRUE)

mart <- useMart('ENSEMBL_MART_ENSEMBL')
mart <- useDataset('hsapiens_gene_ensembl', mart)

annotLookup <- getBM(
  mart = mart,
  attributes = c(
    'hgnc_symbol',
    'ensembl_gene_id',
    'gene_biotype'),
  uniqueRows = TRUE)
head(annotLookup)

```

```

##  hgnc_symbol ensembl_gene_id  gene_biotype
## 1      MT-TF ENSG00000210049      Mt_tRNA
## 2      MT-RNR1 ENSG00000211459      Mt_rRNA
## 3      MT-TV ENSG00000210077      Mt_tRNA
## 4      MT-RNR2 ENSG00000210082      Mt_rRNA
## 5      MT-TL1 ENSG00000209082      Mt_tRNA
## 6      MT-ND1 ENSG00000198888 protein_coding

```

```

filt.annotLookup <- subset(annotLookup, hgnc_symbol != '')

matched.cnv.genes <- filt.annotLookup[filt.annotLookup$hgnc_symbol %in% cnv_final$GeneSymbol, ]
cnv_gene_raw <- which(genes.arr %in% matched.cnv.genes$ensembl_gene_id)
gene.exp$genes <- genes.arr
cnv_gene_count <- gene.exp[cnv_gene_raw,]

```

So after all above steps, we have significant genes from gene expression analysis on each stage, and genes, which have independent recurrent CNV.

Feature selection

Here we used lasso regression as feature selection method with penalty score equal to 0.01.

```

cnv_genes <- cnv_gene_count$genes
stage4_genes <- sig_raw_count_4$gene
stage3_genes <- sig_raw_count_3$gene
stage2_genes <- sig_raw_count_2$gene

concat_genes <- c(cnv_genes,stage2_genes,stage3_genes,stage4_genes)

merged_genes_count <- gene.exp[gene.exp$genes %in% concat_genes,]
rownames(merged_genes_count) <- merged_genes_count$genes
genes <- merged_genes_count$genes
merged_genes_count$genes <- NULL
t_merged_genes_count <- t(merged_genes_count)

fit <- glmnet(t_merged_genes_count, clin.data$OS, alpha = 1, lambda = 0.01)
selected_features <- which(coef(fit, s = 0.01) != 0)
lasso_selected_genes <- gene.exp[selected_features,]
write.table(lasso_selected_genes, file = "/data/gpfs-1/users/phgi10_c/work/Uni/MultiOmic_Datascience/results/gene_exp_analysis/lasso_selected_count.csv", sep = "\t", quote = FALSE, row.names = FALSE)

```

Functional analysis

```

selected_gene_df <- filt.annotLookup[filt.annotLookup$ensembl_gene_id %in% lasso_selected_genes$genes, ]
write.table(selected_gene_df, file = "/data/gpfs-1/users/phgi10_c/work/Uni/MultiOmic_Datascience/results/gene_exp_analysis/gene_annotation.csv", sep = "\t", quote = FALSE, row.names = FALSE)
all_genes_df <- filt.annotLookup[filt.annotLookup$ensembl_gene_id %in% concat_genes, ]
write.table(all_genes_df, file = "/data/gpfs-1/users/phgi10_c/work/Uni/MultiOmic_Datascience/results/gene_exp_analysis/all_gene_annotation.csv", sep = "\t", quote = FALSE, row.names = FALSE)

```