**Lab 4**. Introduction to gene expression data Part 2: Differential Expression. We will use the major depressive disorder (MDD) RNA-Seq data and some basic statistics to identify genes that are differentially expressed between phenotype groups (*e.g.*, between disease and healthy control).

**A**. Prepare Data. Load the RNA-Seq and clinical data from lab 3. Repeat last week’s preprocessing: quantile normalization and log2 transformation.

**1.** How many genes are in the data?

[in] dim(mddExprData\_quantileLog2)

[out]8923 157

So 8923 genes and 157 samples.

# load gene expression data

load("sense.filtered.cpm.Rdata")

# load phenotype (mdd/hc) data

subject.attrs <- read.csv("Demographic\_symptom.csv",

stringsAsFactors = FALSE)

library(dplyr)

# grab intersecting X (subject ids) and Diag (Diagnosis) from columns

phenos.df <- subject.attrs %>%

filter(X %in% colnames(sense.filtered.cpm)) %>%

dplyr::select(X, Diag)

mddPheno <- as.factor(phenos.df$Diag)

# Normalized and transform

library(preprocessCore)

mddExprData\_quantile <- normalize.quantiles(sense.filtered.cpm)

mddExprData\_quantileLog2 <- log2(mddExprData\_quantile)

# attach phenotype names and gene names to data

colnames(mddExprData\_quantileLog2) <- mddPheno

rownames(mddExprData\_quantileLog2) <- rownames(sense.filtered.cpm)

**B**. Filter noise genes. There are many genes in the data that are noise or irrelevant to the disease study at hand. Furthermore, these noise genes waste computational time during analysis. The coefficient of variation (CoV) of a gene (call it x) is the ratio cv(x)=sd(x)/abs(mean(x)), where sd is the standard deviation of the gene expression. A small CoV may mean the experimental effect size is large compared to the measurement uncertainty. The code below uses CoV to filter the genes. Note CoV does not use phenotype information, so we don’t have to worry about biasing the differential expression analysis.

**2.** Add the following code. What does it do? Specifically, what is apply doing (use ?apply)? How many genes are there now?

apply is processing the data by “applying” a function to each value. In this case. In this case it is getting the covariance by getting the standard deviation of the gene across samples and dividing by the mean of that gene across samples. This gives us an idea of if the gene is actually different across samples. If there is no difference or very little difference then we can probably throw them out as they contain little to no information. After that it does the same thing but with standard deviation instead of covariance.

After removing stuff we end up with 5587 genes:

[in] dim(GxS.covfilter)

[out]5587 157

# coefficient of variation filter sd(x)/abs(mean(x))

CoV\_values <- apply(mddExprData\_quantileLog2,1,

function(x) {sd(x)/abs(mean(x))})

# smaller threshold, the higher the experimental effect relative to the

# measurement precision

sum(CoV\_values<.045)

**# there is one gene that has 0 variation -- remove**

**sd\_values <- apply(mddExprData\_quantileLog2,1, function(x) {sd(x)})**

rownames(mddExprData\_quantileLog2)[sd\_values==0]

# filter the data matrix

GxS.covfilter <- mddExprData\_quantileLog2[CoV\_values<.045 **& sd\_values>0**,]

dim(GxS.covfilter)

**C**. Using the t-test to calculate differential expression. The following equation is the two-sample t-test with unequal variances for gene x with a vector of expression levels for groups 1 and 2 (subscripts). The numerator is the difference of the mean expression of gene x between groups 1 and 2. And the denominator involves the pooled deviation from the means of each group, with sizes n and m.

Before we compute the t-test, we need to create a factor variable for our two phenotype groups. A factor is a special R data type: it is a categorical variable with discrete states (levels) that do not necessarily have a meaningful order.

**3**. Add the following code. What are the levels and how many are there?

[in] levels(pheno.factor)

[out] "HC" "MDD"

2 levels: HC, MDD

# convert phenotype

pheno.factor <- as.factor(colnames(GxS.covfilter))

pheno.factor

str(pheno.factor)

levels(pheno.factor)

There are a few interfaces for calculating the t-test in R, which we now try on the second gene in the data (myrow=2 of the data):

myrow <- 2 # first pick a gene row index to test

mygene<-rownames(GxS.covfilter)[myrow]

mygene

**4.** Run the following implementations of the t-test for this gene. Show the output and discuss how implementations a and b differ (use ?t.test). Discuss the evidence for whether this gene is differentially expressed.

# a. traditional R interface

mdd <- GxS.covfilter[myrow,pheno.factor=="MDD"]

hc <- GxS.covfilter[myrow,pheno.factor=="HC"]

t.result <- t.test(mdd,hc)

t.result

[out]

Welch Two Sample t-test

data: mdd and hc

t = 1.4367, df = 154.9, p-value = 0.1528

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

-0.01003427 0.06355762

sample estimates:

mean of x mean of y

5.590087 5.563325

# b. formula interface ~ saves a step

t.result <- t.test(GxS.covfilter[myrow,] ~ pheno.factor)

t.result

t.result$p.value

t.result$statistic

[out]

Welch Two Sample t-test

data: GxS.covfilter[myrow, ] by pheno.factor

t = -1.4367, df = 154.9, p-value = 0.1528

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

-0.06355762 0.01003427

sample estimates:

mean in group HC mean in group MDD

5.563325 5.590087

[in] p.result

[out] 0.1528184

[in] t.statistic

[out] -1.436705

A uses x, y values and B uses a formula. The order it inputs is flipped between the two. So the x value in A is the y value in B. This is why the results are different. The p value is decently sized so it isn’t unrealistic to assume that the gene is not differentially expressed.

**5.** Plot the data. What gene name was used? Would you say this gene is differentially expressed? Why/why not?

gene name: AAK1

No, as the means are very near and there is a lot of overlap.

# plot the data

library(ggplot2)

# create data frame for gene

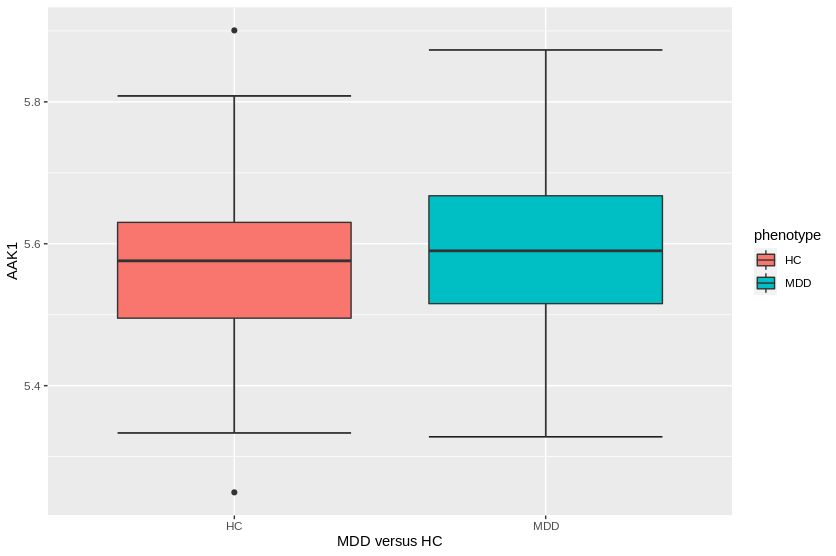
mygene.data.df <- data.frame(gene=GxS.covfilter[myrow,],phenotype=pheno.factor)

# boxplot

p <- ggplot(mygene.data.df, aes(x=phenotype, y=gene, fill=phenotype)) + stat\_boxplot(geom ='errorbar') + geom\_boxplot()

p <- p + xlab("MDD versus HC") + ylab(mygene)

p



**D**. Calculate the t-test for all filtered genes. We need to repeat the t-test for all genes in the data, so we want to choose the fastest and cleanest way of the above implementations. I argue the formula interface is the simplest. In addition we want to create a results function for a given gene that we can use to automate the analysis of all genes in a for loop. Use the following function in your script to do a t-test for a gene with row index i.

# Put it all together into a function to run in loop.

# First write a function that computes t-test for one gene.

# i is the data row for the gene

ttest\_fn <- function(i){

mygene <- rownames(GxS.covfilter)[i]

t.result <- t.test(GxS.covfilter[i,] ~ pheno.factor)

tstat <- t.result$statistic

pval <- t.result$p.value

# return vector of three things for each gene

c(mygene, tstat, pval)

}

Test the function on the second gene using the following command.

**6.** Show the output and explain what the values represent.

ttest\_fn(2)

"AAK1" "-1.43670544207136" "0.152818373849126"

gene name, t-value, p-value respectively.

The following code uses a for loop to apply the user-defined ttest\_fn function above to all genes. Then we sort the results by p-value

**7.** Show the top 10 gene results based on smallest p-values.

# initialize an empty matrix to store the results

ttest\_allgene.mat <- matrix(0,nrow=nrow(GxS.covfilter), ncol=3)

# run analysis on all gene rows

for (i in 1:nrow(GxS.covfilter)){

ttest\_allgene.mat[i,] <- ttest\_fn(i)

}

# convert matrix to data frame and colnames

ttest\_allgene.df <- data.frame(ttest\_allgene.mat)

colnames(ttest\_allgene.df) <- c("gene ", "t.stat", "p.val")

**library(dplyr) # sort based on p-value**

ttest\_allgene.sorted <- ttest\_allgene.df **%>%**

mutate\_at("p.val", as.character) **%>%**

mutate\_at("p.val", as.numeric) **%>%**

arrange(p.val) # sort

ttest\_allgene.sorted[1:10,] # look at top 10

[out]

gene t.stat p.val

1 MDGA1 -4.60305377574858 8.769336e-06

2 ZDHHC20 -4.14665384310082 5.560038e-05

3 NPFF 3.92088360163192 1.322952e-04

4 ARFGAP1 -3.84850546153888 1.745521e-04

5 FAM138A 3.81609056502266 1.977449e-04

6 IRF2BPL -3.80484711944538 2.037987e-04

7 UBD -3.77495951345364 2.295039e-04

8 BCL2L12 -3.70907581977241 2.899766e-04

9 KANTR 3.69624774873673 3.048811e-04

10 CBL -3.694547522781 3.056099e-04

**8.** Add a boxplot for the top gene using the code from C5. First use the following code to specify the top gene for plotting:

# find data row index of top gene name

myrow <- which(ttest\_allgene.df$gene=="**ENTER TOP GENE NAME HERE**")

mygene<-rownames(GxS.covfilter)[myrow]

[in]

myrow <- which(ttest\_allgene.df$gene=="MDGA1")

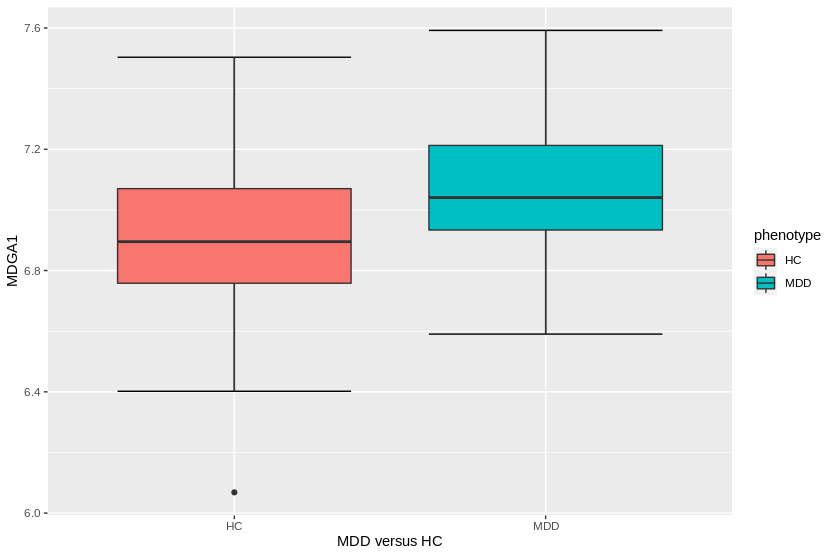
mygene<-rownames(GxS.covfilter)[myrow]

mygene.data.df <- data.frame(gene=GxS.covfilter[myrow,],phenotype=pheno.factor)

p <- ggplot(mygene.data.df, aes(x=phenotype, y=gene, fill=phenotype)) + stat\_boxplot(geom ='errorbar') + geom\_boxplot()

p <- p + xlab("MDD versus HC") + ylab(mygene)

p



**E**. Interpretation of top genes. Look up the top gene in NCBI or Google.

**https://www.ncbi.nlm.nih.gov/gene/266727**

**9.** What if any is the biological evidence for the relevance of the top gene to major depressive disorder?

according to the link above the gene MDGA1 is linked to other neurological disorders

so it is very likely it could have some association with depression disorder. Given the very small p-value this conclusion is likely true.

<https://www.ncbi.nlm.nih.gov/gene/>

Use the following code to print the top-200 gene list.

top\_cutoff <- 200

top\_genes <- as.character(ttest\_allgene.sorted[1:top\_cutoff,1])

write.table(top\_genes, sep="\t", file="", quote=F, row.names=F, col.names=F)

Paste the list into the following MSigDB (molecular signatures database) gene set enrichment analysis (GSEA) tool. You will need to provide an email address to register. Go to Investigate Gene Sets, paste your list of gene names into the Input Box, check Reactome, and click Compute Overlaps.

**10.** What is the top enriched geneset/pathway and what genes from the top list are in it (there is a “Save to: Text” link)?

REACTOME\_POST\_TRANSLATIONAL\_PROTEIN\_MO\_MODIFICATION

https://www.gsea-msigdb.org/gsea/msigdb/geneset\_page.jsp?geneSetName=REACTOME\_POST\_TRANSLATIONAL\_PROTEIN\_MODIFICATION

|  |
| --- |
| PSMD1 |
| NUP88 |
| RBBP5 |
| NR1H2 |
| MEN1 |
| TUSC3 |
| DDX58 |
| CDC20 |
| BTBD1 |
| KLHL22 |
| ST3GAL3 |
| TADA3 |
| SUMO3 |
| MITF |
| VDAC3 |
| UBD |
| ARFGAP1 |
| ARFGAP3 |
| COG2 |
| PIGH |
| MDGA1 |
| TNC |

<http://www.broadinstitute.org/gsea/msigdb/annotate.jsp>