Gene Expression Data Analysis and Visualization
410.671
HW #3

1.) Load the golub data ***training set*** in the multtest library. Also load Biobase and annotate libraries, if they are not loaded with the multtest library. Remember that the golub data training set is in the multtest library, so see the help file for information on this data set (2.5 pts)

```
library(Biobase)
library(annotate)
library(golubEsets)
library(multtest)

dat.train = exprs(Golub_Train)
```

2.) Cast the matrix to a data frame and label the gene names as numbers (e.g. "g1","g2",etc). (2.5 pts)

```
dat.train = as.data.frame(dat.train)
dim(dat.train)
[1] 7129   38
rownames(dat.train) = c(1:7129)
```

3.) Get the sample labels (see lecture notes) and set the sample labels to the data frame. (2.5 pts)

```
ann.dat2 = golub.cl     #diagnosis of ALL=0 and AML=1
```

4.) Use the t-test function in the lecture #7 notes and modify it to "wilcox.test" instead of "t.test". Change the "$p.value" argument to "$statistic". Assign the following arguments to the function: (2.5 pts)
exact=F
alternative="two.sided"
correct=T
Run the function on all of the genes in the dataset and save it as "original.wmw.run"

```
t.test.all.genes <- function(x,s1,s2) {
x1 <- x[s1]
x2 <- x[s2]
x1 <- as.numeric(x1)
x2 <- as.numeric(x2)
t.out <- wilcox.test(x1,x2, alternative='two.sided', var.equal =T, exact = F, correct=T)
out = as.numeric(t.out$statistic)
return(out)
```

```
}
original.wmw.run = apply(dat.train,1,t.test.all.genes,s1=ann.dat2==0,s2=ann.dat2==1)
```

5.) Now write a for loop to iterate 500 times, where in each iteration, the columns of the data frame are shuffled (class labels mixed up), the WMW test is calculated on all of the genes, and the maximum test statistic (W) is saved in a list. (5 pts)

```
max.list = vector('numeric',500)
for(i in 1:500) {
dat.col = dat.train[,sample(1:ncol(dat.train))]
dat.max = apply(dat.col,1,t.test.all.genes,s1=ann.dat2==0,s2=ann.dat2==1)
max = max(dat.max)
max.list[i] = max
print(i)
i = i + 1
}   #this took a surprisingly long time to finish! I used the print(i) statement in the code
    #to keep tabs on where the loop was at, since it took over 20 minutes to complete.
```

6.) Once you have the list of maximum test statistics, get the 95% value test statistic. Subset the original.wmw.run list of values with only those that have a higher test statistic than the 95% value that you calculated. Print the gene names and test statistics out. (5 pts)

```
quantile(max.list, .95)
     95%
271.5
> original.wmw.run[original.wmw.run>271.5]
   804  1144  1630  1928  2233  2348  2354  3507  4328  4375  4535  4546  5501  5772  6281  6855
276.0 280.0 280.0 277.0 279.0 272.0 284.0 276.0 276.0 272.0 280.0 275.5 275.0 288.0 280.0 284.0
> gene.max =
c(804,1144,1630,1928,2233,2348,2354,3507,4328,4375,4535,4546,5501,5772,6281,685
5)
dat.train[gene.max,]
HG1612-HT1612_at
J05243_at
L47738_at
M31303_rna1_at
M77142_at
M91432_at
M92287_at
U62136_at
X59417_at
X62535_at
X74262_at
X74801_at
Z15115_at
U22376_cds2_s_at
M31211_s_at
M31523_at
```

7.) Now we want to compare these results to those using the empirical Bayes method in the limma package. Load this library and calculate p-values for the same dataset using the eBayes() function. (5 pts)

```
library(limma)
fit = lmFit(dat.train,design = NULL)
fit = eBayes(fit)
fit$p.value
```

8.) Sort the empirical Bayes p-values and acquire the lowest *n* p-values, where *n* is defined as the number of significant test statistics that you found in problem 6. Intersect the gene names for your two methods and report how many are in common between the two differential expression methods, when choosing the top *n* genes from each set. (2.5 pts)

```
low.val = as.matrix(fit$p.value)
low.val = low.val[order(low.val),,drop=FALSE]
low.val[1:16,]
```

X67247_rnal_at	X56932_at	X63527_at	X69150_at	X55954_at	X03342_at	X16064_at
1.033387e-46	6.483568e-46	8.756082e-46	9.280109e-43	1.430626e-42	2.054954e-42	1.791065e-41
X73460_at	HG2873-HT3017_at	X17206_at	U14969_at	X15940_at	L04483_s_at	M36072_at
8.152816e-41	1.998371e-39	2.327133e-39	1.073449e-38	2.669177e-38	1.204774e-37	1.359039e-37
M60854_at	Z26876_at					
1.381159e-37	2.654292e-37					

Compared to the list of genes discovered in question 6, there are no gene overlaps between the two methods in the 16 top genes I examined.

9.) Finally, compare the results from a Student's t-test with the empirical Bayes method. To do this, first calculate a two sample (two-tailed) Student's t-test on all genes. Make sure that you are running a Student's t-test and not a Welch's t-test. Then extract only those genes with a p-value less than 0.01 from this test. Plot the gene p-values<0.01 for the Student's t-test vs. the same genes in the empirical Bayes method. Make sure to label the axes and title appropriately. (7.5 pts)

```
t.test.all.genes <- function(x,s1,s2) {
x1 <- x[s1]
x2 <- x[s2]
x1 <- as.numeric(x1)
x2 <- as.numeric(x2)
t.out <- t.test(x1,x2, alternative = 'two.sided',var.equal =T)
out = as.numeric(t.out$p.value)
return(out)}
pv = apply(dat.train,1,t.test.all.genes,s1=ann.dat2==0,s2=ann.dat2==1)pv
stt = pv[pv<0.01]
stt =  as.data.frame(stt)
low.val = as.data.frame(low.val)
bayes = low.val[match(rownames(stt),rownames(low.val)),]
```

```
par(mfrow = c(1,2))
plot(stt,xlab='Genes',ylab='P-Value',main = 'Student T-test P-Values for \n the Golub Training Dataset', col = 'red',pch=21,cex=1.5)
plot(bayes, xlab='Genes',ylab='P-Value',main = 'Empirical Bayes P-Values for \nthe Golub Training Dataset', col = 'blue', pch=1,cex=1.5)
```