Julie Garcia
October 30, 2017

Gene Expression Data Analysis and Visualization
410.671
HW #3

All code is in the lecture notes

1.) Load the golub data training set in the multtest library (in R_. 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(multtest)
library(Biobase)
library(annotate)
data(golub)
```

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

```
data <- data.frame(golub)
rownames(data) <-
lapply(rownames(data),FUN=function(x){paste('g',x,sep="")})
```

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

```
data.ann <- golub.cl
```

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"

```
wilcox.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', exact=F,
correct=T)
  out <- as.numeric(t.out$statistic)
  return(out)
}

original.wmw.run <-
apply(data,1,wilcox.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)

Hints:
Use sample() to sample the number of columns
Get the maximum test statistic across all genes with max()

```
length(colnames(data[ann.dat2==0]))
length(colnames(data[ann.dat2==1]))
maxteststats <- list()
iterate <- for (i in 1:500) {
  x1 <- sample(colnames(data), 27)
  x2 <- sample(colnames(data), 11)
  wmw.run <- apply(data,1,wilcox.test.all.genes,s1=x1,s2=x2)
  maxteststats[i] <- max(wmw.run)
}

iterate()
```

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) (Permutation test, not p-value threshold)

```
max <- max(as.numeric(maxteststats))
ninetyfive <- max * .95
top.ninetyfive <- original.wmw.run[original.wmw.run > ninetyfive]
top.ninetyfive
```

```
g23   g55   g66   g96  g126  g127  g158  g172  g174  g182  g202  g204  g207  g232
  249   251   258   274   262   256   246   253   262   261   255   263   246   250
 g239  g246  g253  g259  g283  g286  g297  g307  g313  g314  g323  g329  g335  g344
  247   254   266   266   271   260   257   252   248   246   262   275   254   259
 g345  g357  g394  g395  g399  g422  g462  g479  g489  g490  g494  g515  g522  g523
  274   247   290   250   247   270   259   248   250   248   251   264   257   283
```

```
 g546  g560  g561  g563  g621  g648  g688  g695  g703  g704  g713  g717  g725  g738
  274   262   281   255   269   271   258   251   285   273   266   283   252   271
 g746  g763  g785  g801  g807  g811  g835  g838  g839  g849  g862  g866  g922  g963
  277   248   256   263   248   249   272   283   272   272   259   269   272   250
 g984 g1006 g1014 g1019 g1037 g1042 g1045 g1060 g1070 g1081 g1086 g1101 g1145 g1162
  283   275   259   254   281   284   270   264   260   249   278   258   262   261
g1202 g1225 g1253 g1271 g1293 g1316 g1327 g1334 g1368 g1381 g1445 g1453 g1455 g1456
  256   254   255   268   255   255   267   266   279   248   263   254   266   261
g1459 g1474 g1524 g1542 g1559 g1564 g1585 g1598 g1610 g1616 g1638 g1640 g1642 g1649
  250   250   282   266   260   246   267   275   249   258   262   265   265   246
g1653 g1671 g1691 g1696 g1723 g1732 g1766 g1773 g1805 g1811 g1817 g1834 g1856 g1869
  259   254   266   250   256   264   252   247   249   284   272   290   263   272
g1882 g1883 g1903 g1909 g1916 g1920 g1926 g1939 g1944 g1947 g1948 g1959 g1963 g1978
  260   281   253   275   273   274   248   268   249   254   252   272   250   271
g1993 g1995 g2002 g2020 g2087 g2105 g2110 g2122 g2179 g2180 g2208 g2213 g2216 g2235
  246   287   283   265   247   252   249   270   267   259   248   252   249   252
g2244 g2262 g2265 g2266 g2289 g2297 g2302 g2307 g2313 g2347 g2356 g2386 g2402 g2410
  246   247   259   272   274   257   254   260   265   250   263   288   262   256
g2418 g2430 g2438 g2466 g2489 g2593 g2616 g2627 g2645 g2673 g2686 g2702 g2736 g2753
  279   259   262   259   288   258   270   253   272   250   253   279   261   260
g2786 g2801 g2803 g2829 g2851 g2860 g2879 g2939 g2950 g2955 g2985 g3046
  246   274   247   273   281   272   272   291   258   267   252   276
```

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)
help("ebayes")

design <-cbind(Grp1=1,Grp2vs1=c(rep(1,27),rep(0,11)))
fit <- lmFit(data,design)
fit <- eBayes(fit)
pv <- fit$p.value[,2]
```

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)

```
sortedps <- sort(pv)
n <- length(top.ninetyfive)
lowest.n <- sortedps[1:n]
lowest.n
common <- intersect(names(lowest.n), names(top.ninetyfive))
length(common)

[1] 95
```

95 genes are in common.

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', exact=F, correct=T)
  out <- as.numeric(t.out$p.value)
  return(out)
}

students <-
apply(data,1,t.test.all.genes,s1=ann.dat2==0,s2=ann.dat2==1)
students.lessthan <- students[students < 0.01]
pv[names(students.lessthan)]
plot(pv[names(students.lessthan)],students.lessthan,
     xlab='empirical Bayes',ylab='Student's t-test',
     main='P-value distribution comparison',
     cex=0.5,col=3,pch=15)
```