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

```
>library(Biobase)
>library(annotate)
>library(golubEsets)
>library(multtest)
>data(golub)
```

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

```
>seq<- seq(1:3051)
>rown<-paste("g", seq, sep="")
>dat<-as.data.frame(golub, row.names = rown)
```

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

```
>colnames(dat)<-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: 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)
        w.out<-wilcox.test(x1,x2, exact=F, alternative="two.sided", correct=T)
        out<-as.numeric(w.out$statistic)
        return(out)
}

>original.wmw.run<-apply(dat, 1, wilcox.test.all.genes, s1=golub.cl==0,
s2=golub.cl==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.

Hints:

Use sample() to sample the number of columns

Get the maximum test statistic across all genes with max()

```
>set.seed(123)
>mts<-c()
>i<-0

>for (i in 1:500){
        mix_dat<-sample(dat)
        shuffled.wmw<-apply(mix_dat, 1, wilcox.test.all.genes, s1=golub.cl==0,
        s2=golub.cl==1)
        mts<-c(mts,(max(shuffled.wmw)))
}
```

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.

```
>quantile(mts, c(.95))
#264

>TSIndex<-which(original.wmv.run>264)
>original.wmv.run[TSIndex]
#76 total genes
```

```
> original.wmv.run[TSIndex]
  g96  g253  g259  g283  g329  g345  g394  g422  g523  g546  g561  g621  g648  g703  g704  g713
  274   266   266   271   275   274   290   270   283   274   281   269   271   285   273   266
 g717  g738  g746  g835  g838  g839  g849  g866  g922  g984 g1006 g1037 g1042 g1045 g1086 g1271
  283   271   277   272   283   272   272   269   272   283   275   281   284   270   278   268
g1327 g1334 g1368 g1455 g1524 g1542 g1585 g1598 g1640 g1642 g1691 g1811 g1817 g1834 g1869 g1883
  267   266   279   266   282   266   267   275   265   265   266   284   272   290   272   281
g1909 g1916 g1920 g1939 g1959 g1978 g1995 g2002 g2020 g2122 g2179 g2266 g2289 g2313 g2386 g2418
  275   273   274   268   272   271   287   283   265   270   267   272   274   265   288   279
g2489 g2616 g2645 g2702 g2801 g2829 g2851 g2860 g2879 g2939 g2955 g3046
  288   270   272   279   274   273   281   272   272   291   267   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.

```
> library(limma)

> zeros<-golub.cl[1:27]
> ones<-golub.cl[28:38]

>design<-cbind(Grp1=1,Grp2vs1=c(rep(1,length(zeros)),rep(0,length(ones))))
```

```
>fit<- lmFit(dat,design)
>fit<-eBayes(fit)

#p-values in: 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.

```
>sort(fit$p.value[,2])

> lowestBayes<-sort(fit$p.value[,2])
> lowestBayes[1:76] #since 76 top n genes

#intersection of:
#lowestBayes[1:76] and original.wmv.run[TSIndex]

>nBayes<-as.data.frame(lowestBayes[1:76])
> nwmv<- as.data.frame(original.wmv.run[TSIndex])

> intersect(row.names(nBayes),row.names(nwmv))
 [1] "g2489" "g1995" "g394"  "g2939" "g717"  "g1042" "g523"  "g2702" "g1037"
"g1811" "g1883" "g2386" "g849"  "g746"  "g1834" "g2266" "g561"  "g1524"
"g2289"
[20] "g2851" "g738"  "g1978" "g2801" "g2616" "g2418" "g2860" "g1909"
"g2002" "g1271"

#There are 29 top genes in common between the two differential expression
methods.
```

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.

Paste all information into a PDF.

```
>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=TRUE)
        out<-as.numeric(t.out$p.value)
```

```
        return(out)
}

>pv<-apply(dat, 1, t.test.all.genes, s1=golub.cl==0, s2=golub.cl==1)

>PVIndex<-which(pv<.01)
#pv values < .01 in Student's t-test = pv[PVIndex]
#Bayes pv values for same= fit$p.value[,2][PVIndex]

> plot(as.numeric(fit$p.value[,2][PVIndex]), as.numeric(pv[PVIndex]),
xlab="Empirical Bayes", ylab="Student's T-Test", main="P-value Distribution
Comparison Using Golub Data Set", cex=.5, col=3, pch=15)
```