Home work 3

Kevin Chen

7/23/2020

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(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)

```
golub.df<-as.data.frame(golub)
gene.names<-paste0("g",1:nrow(golub.df))
rownames(golub.df)<-gene.names
```

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

```
colnames(golub.df)<-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 “ $p.\ value$ " $argumentto$ " 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",exact =F,correct=T)
    out <- as.numeric(t.out$statistic)
    return(out)
}
# s1 and s2 are dimensions of the two samples
# run function on each gene in the data frame
original.wmw.run <- apply(golub.df,1,t.test.all.genes,s1=colnames(golub.df)=="0",s2=colnames(golub.df)=="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()

```
max.list<-c()
set.seed(1)
for(i in 1:500){
  tmp.lab<-sample(colnames(golub.df),size=ncol(golub.df),replace=F)
  tmp.stats<-apply(golub.df,1,t.test.all.genes,s1=tmp.lab=="0",s2=tmp.lab=="1")
  max.list<-c(max.list,max(abs(tmp.stats)))
}
```

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)

```
cutoff<-quantile(max.list,prob=0.95)
sig.genes<-original.wmw.run[abs(original.wmw.run)>cutoff]
sig.genes
```

```
##    g96  g283  g329  g345  g394  g422  g523  g546  g561  g621  g648  g703  g704
##    274   271   275   274   290   270   283   274   281   269   271   285   273
##   g717  g738  g746  g835  g838  g839  g849  g866  g922  g984 g1006 g1037 g1042
##    283   271   277   272   283   272   272   269   272   283   275   281   284
## g1045 g1086 g1271 g1327 g1368 g1524 g1585 g1598 g1811 g1817 g1834 g1869 g1883
##    270   278   268   267   279   282   267   275   284   272   290   272   281
## g1909 g1916 g1920 g1939 g1959 g1978 g1995 g2002 g2122 g2179 g2266 g2289 g2386
##    275   273   274   268   272   271   287   283   270   267   272   274   288
## g2418 g2489 g2616 g2645 g2702 g2801 g2829 g2851 g2860 g2879 g2939 g2955 g3046
##    279   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. (5 pts)

```
library(limma)

design <-cbind(Grp1=1,Grp2vs1=as.numeric(colnames(golub.df)))
fit <- lmFit(golub.df,design)
fit <- eBayes(fit)
ebayes.p.values<-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)

```
n<-length(sig.genes)
ebayes.p.values<-sort(ebayes.p.values)
lowest.n.ebayes<-ebayes.p.values[1:n]
shared.genes<-intersect(names(sig.genes),names(lowest.n.ebayes))
print(paste0("when choosing the top ",n," genes from each set, ",length(shared.genes)," are in common betwee
n the 2 differential expression methods"))
```

```
## [1] "when choosing the top 65 genes from each set, 21 are in common between the 2 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. (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$p.value)
return(out)
}
# s1 and s2 are dimensions of the two samples
# run function on each gene in the data frame
pv <- apply(golub.df,1,t.test.all.genes,s1=colnames(golub.df)=="0",s2=colnames(golub.df)=="1")
student.pv.filtered<-pv[pv<0.01]
target.genes<-names(student.pv.filtered)
ebayes.pv.filtered<-ebayes.p.values[target.genes]
temp<-data.frame("genes"=target.genes,"student"=student.pv.filtered,"empirical"=ebayes.pv.filtered)
plot(temp$empirical,temp$student,main="comparison of p values between student's t test\nand empirical bayes"
,xlab="Empirical Bayes",ylab="Student's t test")
```

comparison of p values between student's t test and empirical bayes